DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943

DECOMPOSITION OF DUAL-ANGULAR INTEGER PROGRAMS, WITH APPLICATIONS TO MULTI-STAGE CAPITAL BUDGETING

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF OPERATIONS RESEARCH

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Kevin John Reardon

July 1974

T170828

Thesis
R2545

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93940

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Approved for the University Committee
on Graduate Studies:

ACKNOWLEDGMENTS

I extend my sincere appreciation to Professor Frederick S. Hillier who introduced me to the fields of integer programming and capital budgeting, and who served as my principal research advisor. Professor Hillier's active interest, constant encouragement and expert guidance contributed substantially to the successful completion of this dissertation. I am also indebted to Dr. John A. Tomlin for his many helpful suggestions on the integer programming portion of my work, and to Professor Gerald J. Lieberman for his interest and guidance during my four years at Stanford.

The opportunity and support for my graduate education was provided by the Office of the Chief of Naval Operations under the Junior Line Officer Advanced Educational Program (Burke Program), and is gratefully acknowledged. Additional support during the final year of my studies was funded by the Office of Naval Research and the National Science Foundation.

I have become increasingly aware of the importance of my parents' advice and encouragement over the years, and gratefully share credit with them for this dissertation. To my wife, Sue, who put up with odd study hours and unmowed lawns, and whose loving optimism helped keep me going, I owe much more than words can express.

A special note of thanks goes to Miss Gail Lemmond for her excellent job of drafting figures and typing from an occasionally illegible manuscript.

TABLE OF CONTENTS

CHAPTER		PAGE
	ACKNOWLEDGMENTS.	iii
1	INTRODUCTION AND DISCUSSION OF CAPITAL BUDGETING	1
	1.1 Historical Perspective.	2
	1.2 The Capital Rationing Problem	6
	1.3 Capital Budgeting with Borrowing.	10
	1.4 Chance-Constrained Programming Formulation.	12
2	THE MULTI-STAGE MODEL.	17
	2.1 Basis for the Model	17
	2.2 The Basic Two-Stage Model	18
	2.3 Extension for Uncertain Future Opportunities. . . .	23
3	STRUCTURE AND SOLUTION OF THE MODEL.	29
	3.1 Size and Structure Aspects.	29
	3.2 Inclusion of Borrowing/Lending and Fractional Investments	35
	3.3 Summary	40
4	THE DECOMPOSITION METHOD: GENERAL DISCUSSION.	42
	4.1 Applications of the Method.	42
	4.2 Motivation for the Method	44
	4.3 Description of the Method	46
5	THE DECOMPOSITION METHOD: IMPLEMENTATION.	51
	5.1 Introduction.	51
	5.2 Penalties	52
	5.3 Surrogate Constraints	59
	5.4 Steps of the Algorithm.	62
	5.5 Possible Modifications to the Algorithm	72
6	COMPUTATIONAL RESULTS.	75
	6.1 Computer Code	75
	6.2 Description of Testing Procedure.	77
	6.3 Test Problem Number One	79
	6.4 Test Problem Number Two	83
	6.5 Test Problems Three and Four.	84
	6.6 Test Problems Five and Six.	91
	6.7 Test Problem Number Seven	91
	6.8 Schrage's Test Problem.	96
	6.9 Summary of Computational Results.	100

CHAPTER PAGE

7 CONCLUSIONS AND AREAS FOR FUTURE RESEARCH. 103
7.1 Capital Budgeting Formulation 103
7.2 Subproblem-Decomposition Method 105
7.3 Areas for Future Research 106

APPENDIX 109

REFERENCES 122

CHAPTER 1

INTRODUCTION AND DISCUSSION OF CAPITAL BUDGETING

One of the most important recurring decisions for any economic unit, public or private, is that of allocating its limited financial resources in a manner which best supports the attainment of its goals. Nearly always, such decisions must be made in an environment characterized by incomplete information, uncertainty, complex interactions among activities, imperfect capital markets, and many other complicating factors. The problem is compounded by the fact that most organizations in government and private business continue to perform their functions over a number of years, and must therefore consider the possible future impacts of their decisions. In fact, it is often necessary not only to resolve those issues on which immediate action is required, but to integrate such current decisions into a strategic plan which applies far into the future.

During the past quarter-century, managerial economists have given considerable attention to this "capital budgeting" problem, and have greatly improved the methods used to deal with it. The theory of capital budgeting is by no means complete, however, and there remain many unanswered questions of practical significance. The development of techniques for constructing and solving mathematical models of complex decision processes has allowed explicit consideration of interactions among investment alternatives and other complicating factors. Such mathematical programming techniques have provided a

more theoretically-sound basis for decision making than was previously available, and the computational efficiency with which they may often be solved shows great promise for the resolution of large-scale capital budgeting problems.

This dissertation describes and comments upon several formulations of the capital budgeting problem which have appeared in the literature of mathematical programming, and presents an extended multi-stage model for the selection and timing of investment proposals whose future availability is uncertain. The model is developed in the context of pure and mixed integer linear programming, with particular attention given to the special structure of its constraint set. We then present an integer linear programming method designed to exploit special structures of this type, commonly known as "dual-angular", by decomposition into groups of smaller subproblems. (Such dual-angular constraint matrices also arise in a number of other applications not directly related to multi-stage capital budgeting.) The decomposition method, implemented within the framework of a penalty-based zero-one integer programming algorithm, has enjoyed considerable success in reducing the effort required for solving a variety of test problems. A detailed discussion of this specific implementation and of its associated computational experience is also presented.

1.1. Historical Perspective

Prior to the publication of Lorie and Savage [1955], most theoretical techniques for selection of investment proposals were

concerned with the analysis of individual projects. Acceptance or rejection of any given opportunity was based mainly on the value of some measure of merit, which was calculated with limited consideration of physical and budgetary interactions with other opportunities and the current operations of the firm.[1] Such a procedure can result in an optimum choice of projects only if all those under consideration are completely independent of one another.[2] This precludes such commonplace interrelationships as competition for the same scarce resources (wealth, manpower, customers, etc.) or the sharing of costs for new facilities required by two or more projects.

A classic example of such techniques for evaluating individual projects is the payback criterion. Using this approach, a proposal is deemed acceptable if its total earnings meet or exceed the required initial capital outlay within a specified period of time. The determination of the "payback period" is rather arbitrary, with management often requiring that those projects which it considers more "risky" than others be repaid in a shorter period of time. In addition, different types of investments are frequently assigned different basic payback guidelines which are then "risk-adjusted". Some of the disadvantages associated with this approach are: (1) cash flows (positive or negative) which occur after the specified payback period are completely disregarded in the evaluation; (2) the salvage value of a project which may be terminated beyond its payback limit is not considered; (3) the value of a dollar received at any time

[1] See Dean [1951] and Klevorick [1969] for additional discussions.

[2] This has been discussed by Lorie and Savage [1955], Hirshleifer [1958], Weingartner [1967], and Klevorick [1969].

during the payback period is considered to be a constant, even though the value of money to the firm may vary within that period; (4) if different payback requirements are levied on different projects, there arises the situation wherein a dollar received from project A after its payback period has expired is given no value, but the same dollar received from project B at the same time is assigned a positive utility; and (5) the criterion provides little guidance when budgetary constraints are imposed on several periods.

While not advocating payback as a criterion for selecting capital investments, Weingartner [1971] has discussed in detail its apparent attractiveness to businessmen. Payback was presented variously as a measure of rate of return, as a form of break-even analysis, as a measure of the "liquidity" of a capital asset, and as a measure of the rate of resolution of uncertainty. Finally, it was concluded that payback was an oversimplification in each of the above cases, but that further study of its popularity might be a prerequisite to the development of superior alternatives.

Another well-known technique for evaluating individual projects is the internal rate of return rule. This rule requires the calculation for each project of that rate, r, which would cause the project's cash flows over future years to sum to zero if they were first discounted by r. That is, a solution, r, is sought for the polynomial equation

$$(P)\quad a_1 + \frac{a_2}{1+r} + \frac{a_3}{(1+r)^2} + \cdots + \frac{a_T}{(1+r)^{T-1}} = 0 \quad ,$$

where a_i is the net cash flow associated with the project in period i. The procedure then requires that the proposals be ranked in order of decreasing internal rate of return, and that all those with rates exceeding a predetermined "rejection rate" (e.g., "cost of capital") be accepted. This method suffers from several shortcomings: (1) the internal rate of return as defined above may not be unique, or the only roots of the polynomial (P) may be complex; (2) the cash flows in every period are given the same relative value in calculating r, even though the relative value of money to the firm may vary from period to period; (3) the "cost of capital" is an extremely elusive figure whose determination has received much attention in the financial literature[3]; and (4) the selection criterion provides insufficient guidance when the choice of projects is restricted by constraints on total cash flow in several periods.

The net present value rule is similar to the internal rate of return rule, except that the former's discount rate is externally-supplied (often by a measure of the "cost of capital") instead of being determined from the values of the cash flow components. In fact, the applicable discount rate may vary from one period to another when using the former rule, whereas the latter uses a uniform rate. This allows for explicit use of information regarding future financial trends; however, the firm is still faced with the difficult task of determining the proper discount factor for each period. Under the assumptions of a perfect capital market and complete independence among projects, use of the borrowing rate for discounting and selection

[3] See Hirshleifer [1958] or Klevorick [1969], for example.

of all projects with positive present values will lead to a solution which maximizes the net present value of the firm. Unfortunately, relaxation of either of these restrictive assumptions requires modification of the selection rules and may well render the rules non-optimal. In particular, if multi-period budgetary constraints are imposed, it becomes theoretically impossible to determine the appropriate discount rates and the aforementioned selection rules would fail even if the rates were available. (For a detailed discussion of the problems involved in searching for the proper discount rates, the interested reader is referred to Hirschleifer [1958].)

1.2. The Capital Rationing Problem

Lorie and Savage [1955] presented a capital budgeting model designed to overcome some of the limitations of the then generally-accepted internal rate of return criterion for project selection.[4] The problem was to select a portfolio of investment projects which would have maximum total present value, subject to upper bounds on total expenditures in several periods and an exogenously-determined value for the firm's cost of capital in each period. Thus, rather than evaluating individual projects on their own merits and making independent investment decisions, the objective was to make a concurrent selection which explicitly considered the budgetary interactions of the available projects. Assuming that all cash flows were known with certainty and that the projects were mutually independent, Lorie and Savage were able to make some progress toward their goal when budgetary

[4] See Dean [1951].

constraints applied in only one or two periods and fractional investments were allowed. Their iterative method was a form of "generalized Lagrange multiplier" technique, as discussed by Everett [1963].

In a highly significant 1962 doctoral dissertation, Weingartner [1967] showed that the Lorie-Savage problem could be formulated as a linear program as follows:

$$
\begin{aligned}
(\text{LP}) \qquad \text{maximize} \quad & \sum_j b_j x_j \,, \\
\text{subject to} \quad & \sum_j a_{jt} x_j \leq M_t, \qquad \text{for} \quad t = 1, \ldots, T, \\
& 0 \leq x_j \leq 1 \qquad \text{for all} \quad j \,,
\end{aligned}
$$

where

x_j is the level of participation in project j,

b_j is the net present value of project j,

a_{jt} is the gross expenditure required for project j in time period t, and

M_t is the spending limit which applies in period t.

Recognition of this formulation allowed application of the powerful theoretical and computational techniques of linear programming. In particular, Weingartner presented an extensive analysis of the economic interpretations of duality theory as applied to (LP) and various extensions.

The extensions to (LP) which were discussed included a change in the objective to maximize the worth of the firm at some future time (the "horizon"), and allowances for borrowing and lending under several

types of capital market imperfections. Although some types of investment opportunities are highly divisible and may therefore be undertaken at levels corresponding to fractional values of their associated decision variables (e.g., borrowing or lending), most capital budgeting decisions are of the accept-reject variety for which indications of fractional participation have little or no meaning. By appending to (LP) the additional constraint

$$(*) \qquad x_j = 0 \text{ or } 1, \qquad \text{for } j \in S,$$

one may exclude the possibility of obtaining such indications for those projects belonging to the set S.

Also, as Weingartner pointed out, this integer linear programming formulation can be used to express various types of project interdependencies by use of additional constraints. For example, the constraint

$$x_k + x_\ell \leq 1 \qquad \text{where } k, \ell \in S$$

together with (*) indicates that projects k and ℓ are mutually exclusive, and

$$x_m - x_n \leq 0, \qquad \text{where } m, n \in S$$

together with (*) indicates that acceptance of project m requires acceptance of project n. Unfortunately, the solution of an integer linear program is a much more difficult task than that of the corresponding

linear program in which the integer restrictions are removed; furthermore, the most successful (enumerative) techniques for integer linear programming are not readily amenable to duality analysis or sensitivity analysis.

Baumol and Quandt [1965] expressed several criticisms of Weingartner's formulation of the capital rationing problem. They considered the most serious shortcoming to be the previously-mentioned difficulty of determining an appropriate discount rate for each period, and indicated that other writers had unjustifiably glossed-over the problem. In addition, they were dissatisfied with the fact that Weingartner's model did not allow net cash inflows from selected projects to be used in the same periods for augmenting the budgeted amounts. Baumol and Quandt argued that a firm should be able to reinvest such funds, and that it should be concerned with maximizing the utility of the funds withdrawn from its portfolio, rather than with maximizing the present value of that portfolio. They further claimed that such an approach would have the additional advantage of obviating the need for determining discount rates for each period, and proposed the following model:

$$\text{maximize} \qquad \sum_t U_t W_t$$

$$\text{subject to} \qquad -\sum_t a_{jt} + W_t \leq M_t , \qquad \text{for } t = 1, \ldots, T,$$

$$x_j \geq 0 , \qquad \text{for all } j,$$

$$W_t \geq 0 , \qquad \text{for all } t,$$

where U_t is the utility of one dollar of withdrawal in period t, W_t is the amount withdrawn in period t, and a_{jt} is the net cash flow from project j in period t.[5]

Weingartner [1966b] responded to the Baumol-Quandt paper with an admission of the difficulty involved with discount rates and a defense of his decision not to allow reinvestment of earnings. He also pointed out several faults in the Baumol-Quandt model, and suggested a third model which sought to maximize dividend growth. Following this exchange, several other papers appeared with suggestions on formulating an appropriate objective function for the capital rationing problem.[6] It appears, however, that the controversy remains unresolved, and quite possibly may never by completely settled in a way that is both theoretically sound and computationally tractable. Meanwhile, advocates of programming approaches to capital budgeting have tended to choose particular objective measures (recognizing their imperfections) and to proceed to develop models and techniques which can improve current practices. The most widely-accepted objective criterion is that of maximizing total net present value from the projects selected, and that is the measure which will be used in the remainder of this study.

1.3. Capital Budgeting with Borrowing

Before proceeding to a dicussion of the incorporation of risk in mathematical programming models and to the presentation of our proposed multi-stage model, we should first show the extension of

[5] The authors stated that the use of a non-linear utility function would add nothing to their argument.

[6] Carleton [1969], Lusztig and Schwab [1968], Mao [1969], and Bernhard [1969].

Weingartner's deterministic formulation to allow for borrowing of funds. This capital rationing model can be readily altered to allow the firm to borrow funds under various types of capital market conditions. The budgetary constraints remain in the model, but are now interpreted as representing internally-generated funds which may be supplemented at a cost from external sources. We can also assume that surplus funds in any period may be lent out at the prevailing lending rate of interest, which may differ from the borrowing rate. Assuming unlimited ability to borrow and lend on one-period contracts at constant rates i_B and i_L, respectively, and using i_B for the discount factor, the model becomes:

$$\text{maximize} \quad \sum_j b_j x_j - i_B \sum_t \frac{u_t}{(1+i_B)^t} + i_L \sum_t \frac{v_t}{(1+i_B)^t} ,$$

$$\text{subject to} \quad \sum_j a_{j1} x_j - u_1 + v_1 \leq M_1 ,$$

$$\sum_j a_{jt} x_j + (1+i_B) u_{t-1} - (1+i_L) v_{t-1} - u_t + v_t \leq M_t ,$$

$$\text{for} \quad t = 2, \ldots, T ,$$

$$0 \leq x_i \leq 1 , \qquad \text{for all} \quad i,$$

$$u_t, v_t \geq 0, \qquad \text{for} \quad t = 1, \ldots, T ,$$

where u_t and v_t are the amounts borrowed and lent, respectively, in period t.

We can introduce various market imperfections by adding additional constraints to the above formulation. For example, the provision for

unlimited borrowing can be replaced by upper bounds, B_t, on the amounts borrowed in each period:

$$u_t \leq B_t, \qquad \text{for } t = 1, \ldots, T.$$

Such restrictions might be applied to a division of a firm by its home office, or may arise for the entire firm due to liquidity considerations or limitations imposed by lending agents.

Another realistic provision we might make is for the borrowing rate to be an increasing function of the amount borrowed. This is easily accomplished in any period by replacing u_t with variables u_{tk}, associating a different interest rate with each value of k, and placing an upper bound on the amount that may be borrowed under each rate. This extension results in a linear program with a separable convex objective, which can be readily solved with the simplex method. Weingartner [1967] has also extended his basic model to consider the amount and timing of equity financing.

1.4. Chance-Constrained Programming Formulation

Up to this point, we have been concerned only with the formulation and analysis of the deterministic capital budgeting problem. We have assumed that the decision-maker knows with complete certainty what projects will be available for selection, what profits and cash-flow patterns will be associated with these projects, what limitations will be applied to expenditures and borrowing, and what the prevailing rates

of interest will be during that portion of the future with which he is concerned. Although the solution of a problem under these assumptions can provide valuable insight into the implications of various constraints and the financial and physical interactions among projects, "real world" decisions require the incorporation of the firm's attitude toward risk -- the possibility that investments may not turn out as well as expected.

In order to incorporate this concept of risk into the analysis, one must first identify those factors which can significantly affect the solution when they are varied within their ranges of possible values. The next step is to estimate probability distributions on these most sensitive variables[7], and to set all others at nominal (perhaps most likely, or mean) values. One can then employ a measure of the desirability of a project portfolio based on the expected results and the attendant risk, and proceed to solve and analyze the problem.

Many different methods exist for decision-making under risk, but the two which have received the most attention in the capital budgeting literature are the expected-utility approach and chance-constrained programming. Maximization of expected utility has been a popular objective among authors in the related area of portfolio selection[8], and has been applied in recent years to capital budgeting by Hillier [1969] and Klevorick [1969]. Its main difficulties lie in the complicated utility functions (cubic, exponential, etc.) which arise, and in the amount of work necessary to adequately determine these functions. The chance-constrained programming approach seeks to avoid these difficulties by using a simpler objective function and incorporating

[7] See Hillier [1963] and [1965].

[8] Markowitz [1959].

risk-aversion into the constraints of the problem. This is done by requiring that any feasible selection of projects have a probability of at least α_i of satisfying the ith constraint.[9] Although several types of objective criteria have been proposed for chance-constrained programming, the most common one is to maximize the expected value of the chosen measure-of-merit, such as net present value.

Assuming a long-run perfect capital market, Naslund [1967, 1971] presented and discussed a chance-constrained capital budgeting model in a manner which closely paralleled Weingartner's deterministic development. He showed how chance-constraints led to the requirement for a specific financial reserve, or "cushion", in each period, with the amount of that reserve depending on the amount of risk considered acceptable. Naslund also developed acceptance criteria for projects and considered the effects of changes in interest rates and variance of returns. The model -- a direct extension of Weingartner's horizon posture model -- was as follows:

$$\text{maximize} \qquad E\Big(\sum_i \hat{d}_i\, x_i + v_T - w_T\Big) ,$$

$$\text{subject to} \qquad \text{Prob}\Big(\sum_i d_{i1}\, x_i + v_1 - w_1 \le D_1\Big) \ge \alpha_1 ,$$

$$\text{Prob}\Big(\sum_{j=1}^{T} \sum_i d_{ij}\, x_i - \sum_{j=1}^{t-1} v_j r + \sum_{j=1}^{t-1} w_j r + v_t - w_t \le \sum_{j=1}^{t} D_j\Big) \ge \alpha_t ,$$

$$\text{for } t = 2, \ldots, T,$$

$$0 \le x_i \le 1 , \qquad \text{for all } i,$$

$$u_t,\ v_t \ge 0 , \qquad \text{for } t = 1, \ldots, T,$$

[9] See Hillier and Lieberman [1967], pp. 536-42, for an introduction to chance-constrained programming.

where

E is the expected value operator,

D_t is the amount of money supplied externally in period t,

d_{it} is the random variable representing net cash flow from project i in period t,

v_t is the amount lent in period t,

w_t is the amount borrowed in period t,

r is the borrowing, lending, and discount rate,

x_i is the fraction of project i undertaken,

$\hat{d}_i$ is the expected horizon value of all post-horizon cash flows from project i,

T is the horizon time.

In this case, the probabilistic budgetary constraints are imposed on the cumulative net cash flows as adjusted for interest received or paid out. It is assumed that borrowing and lending take place on yearly contracts, and that these contracts must be negotiated prior to knowing the outcomes in the respective periods. Therefore, it is not possible to guarantee with certainty that the cash flow constraints will be satisfied under all conditions.

Assuming that the random variables, d_{ij}, are independently normally-distributed with mean μ_{ij} and variance σ^2_{ij}, we can derive a deterministic equivalent of the model wherein all investment and financing decisions are made at the outset (a zero-order decision rule which does not explicitly contain the random elements d_{ij}):

$$\text{maximize} \quad \sum_i \hat{d}_i x_i + v_T - w_T ,$$

$$\text{subject to} \quad \sum_i \mu_{i1} x_i + v_1 - w_1 + \sqrt{\left(\sum_i \sigma_{i1}^2 x_i^2\right)}\, F^{-1}(\alpha_1) \leq D_1$$

$$\sum_{j=1}^{t} \sum_i \mu_{ij} x_i - \sum_{j=1}^{t-1} v_j r + \sum_{j=1}^{t-1} w_j r + v_t - w_t$$

$$+ \sqrt{\left(\sum_i \sigma_{it}^2 x_i^2\right)}\, F^{-1}(\alpha_t) \leq \sum_{j=1}^{t} D_j , \qquad \text{for } t = 2, \ldots, T,$$

$$0 \leq x_i \leq 1 , \qquad \text{for all } i ,$$

$$u_t, v_t \geq 0 , \qquad \text{for } t = 1, \ldots, T$$

where $F(\cdot)$ represents the appropriate distribution function for each constraint.

Had the random variables, d_{ij}, not been independent, the deterministic equivalent constraints would have included covariance terms under their radicals. As Naslund pointed out, the model can easily be extended to allow the D_j to be normally-distributed random variables. Constraints expressing contingency and mutual exclusion relationships may, of course, be added as necessary.

The terms above which involve square roots represent the financial slacks, or "cushions", which must be added to the expected net cash flows as a precaution against the possibility that the actual realized flows might violate the budget constraints. As intuition would indicate, the sizes of these financial slacks are increasing functions of the cash flow variances and the required probabilities, α_i.

CHAPTER 2

THE MULTI-STAGE MODEL

2.1. Basis for the Model

The model which we will propose in this chapter is an extension of the "basic two-stage model" for chance-constrained capital budgeting which was presented in Hillier [1969]. Hillier's model was designed to facilitate a strategy approach to the selection of future projects as well as those currently available. In the two-stage context, certain decisions are made immediately (stage 1) with consideration given to investment opportunities which will become available in the future, and then a second set of decisions will be made in the future (stage 2) after having observed some of the results from the first set. This type of formulation not only can provide guidance on which future investments should be selected, given specific outcomes from the first group, but also can indicate whether the previously selected projects should be reduced, continued, or expanded. In general, the model can be set up to handle any number of stages, but for purposes of exposition we will confine most of our discussion to the two-stage case.

Using the notation of Hillier [1969], the general problem is to determine a decision vector, $\delta = (\delta_1, \ldots, \delta_m)$, whose zero-one elements indicate the rejection or acceptance, respectively, of their associated projects. The determination is to be made so as to maximize the expected net present value of the portfolio, subject to probabilistic constraints on net cash flow in each period and on cumulative net cash flow through

each period. In addition, it is required that δ be a member of the set of feasible decision vectors, designated by S. This set of vectors may be limited by mutual exclusion, or contingency relationships among projects, and may also reflect additional constraints of a budgetary or non-budgetary nature. We further assume that the arguments of the probability functions and the restrictions on S can be expressed as linear inequalities.

In notational form, the problem is to

$$\text{maximize} \qquad E\{P(\delta)\} = \sum_{j=1}^{m} E(P_j)\delta_j \ ,$$

$$\text{subject to} \qquad \text{Prob}\{(NCF)_k \geq L_k\} \geq \alpha_k, \qquad \text{for } k = 1, \ldots, n,$$

$$\text{Prob}\{(CNCF)_k \geq C_k\} \geq \eta_k, \qquad \text{for } k = 2, \ldots, n,$$

$$\delta \in S \ , \ \delta_j = 0 \text{ or } 1 \qquad \text{for } j = 1, \ldots, m,$$

where $P(\delta)$ is the net present value of the portfolio, P_j is the net present value of project j, $(NCF)_k$ is the net cash flow from the projects in period k, $(CNCF)_k$ is the cumulative net cash flow through period k, L_k and C_k are the bounds imposed in period k, and α_k and η_k are the specified risk levels in period k.

2.2. The Basic Two-Stage Model

Hillier's "basic two-stage model", which was mentioned previously, can be presented as a specific form of the general problem stated above.

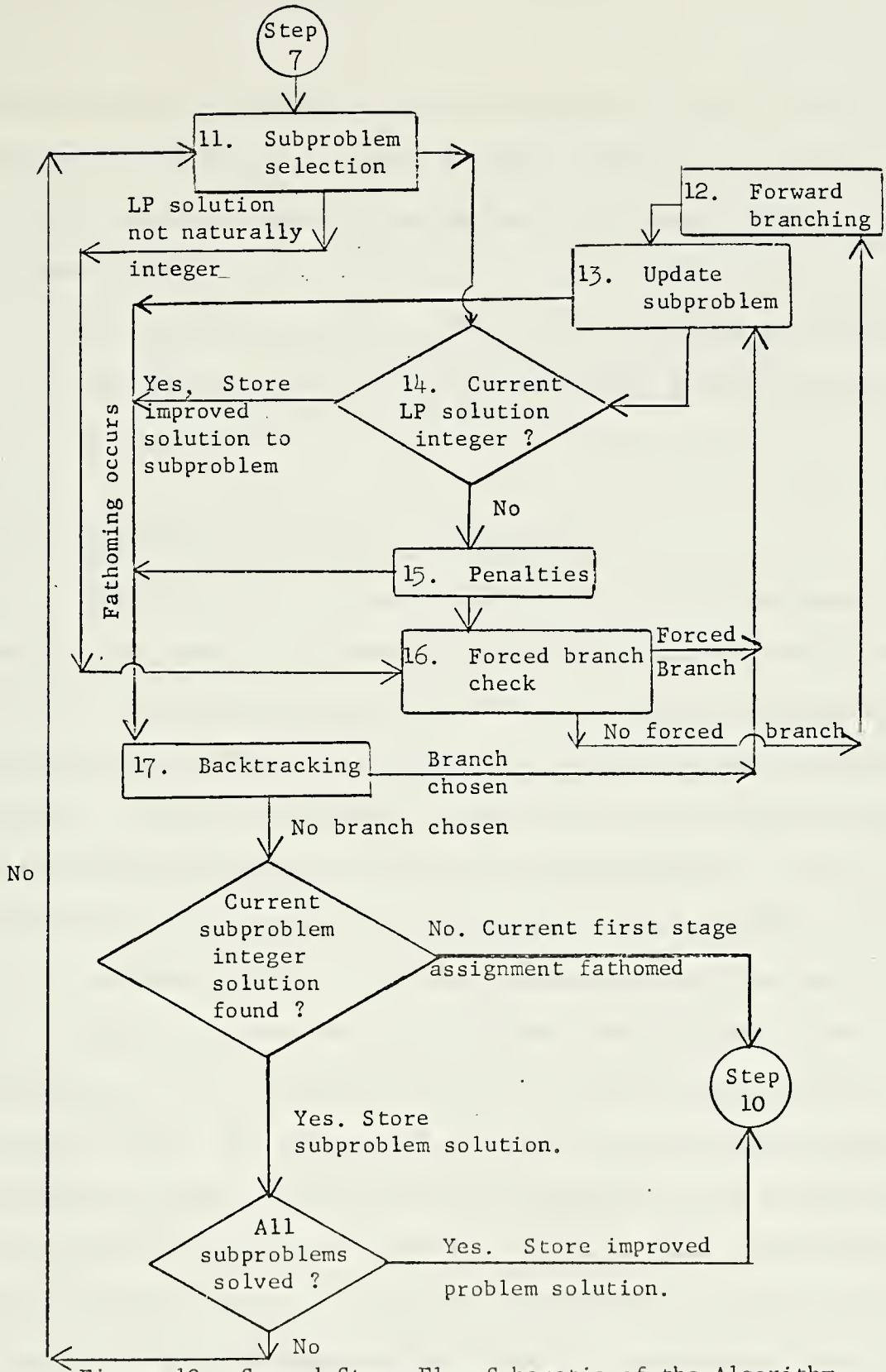


Figure 10. Second Stage Flow Schematic of the Algorithm

associated with the subproblem. If $\ell \neq L$, go to step 11. If $\ell = L$, the assignment of first and second stage variables in S gives an improved integer solution to the problem. Set $z^* = \sum_{\ell=1}^{L} z_\ell^*$, and return to step 10.

If no integer solution to the current subproblem is in storage ($z_\ell^* = -\infty$), the subproblem is integer-infeasible. Return to step 10.

5.5. Possible Modifications to the Algorithm

The literature of integer programming is rich in techniques which may be used to supplement or replace many of those used in the specific implementation described above. It would be well beyond the scope of our discussion to present a listing and discussion of these methods. Since the main body of our contribution is the technique for relaxation to independent subproblems and decomposition, rather than the penalty/surrogate LP approach used to implement this technique, we refer the reader to any of the recent surveys and texts in the field.[29]

There are several tactical considerations involved in the application of our implementation which may lead to reductions in solution times. For example, if the option to use surrogate linear programs is chosen, it may prove more economical to form a new surrogate LP less often than at each iteration of step 6. This would eliminate much of the time spent in setting up the problems and finding initial feasible solutions (simplex method, Phase I). In addition, when a single set of surrogate constraints is used for several iterations,

[29] See, for example, Geoffrion and Marsten [1972] or Garfinkel and Nemhauser [1972].

restarting techniques can be applied to update previous solutions, thereby saving time which would otherwise be required by Phase II of the simplex method. The tradeoff in this approach derives from the fact that "older" surrogate constraints may not be as strong as newly-created ones, thus resulting in reduced fathoming ability.

As mentioned in Section 3, surrogate LP's were used to supplement the fathoming test of step 6, which utilized Gomory penalties for each subproblem. Considerable savings might be realized from eliminating the calculation of subproblem penalties and relying more on the fathoming power of the surrogate linear program and its penalties. In addition, it would be quite easy to apply Balas-type logical tests to the already-available surrogate constraints when working on the assignment of first stage variables. Use of these tests when assigning second stage values would require explicit formation of the surrogate constraints from available data -- not a difficult task.

No attempt was made to identify and utilize any naturally-integer first stage variable values in the subproblems or surrogate LP. Provided all subproblems agree in their assignment of integer values to unfixed first stage variables, it is possible to fix all such variables and proceed to step 11. There is, of course, no guarantee that a naturally-integer first stage assignment in a surrogate LP will be feasible for the individual subproblems, and we conjecture that an attempt to use such assignments will be less successful than one based on subproblem solutions.

As pointed out by Schrage [1973], one of the disadvantages of a decomposition approach such as ours is that the necessity of assigning

values to all first stage variables prior to working on the second stage can lead to wasted time. This can be true if the values of several first stage variables are relatively "unimportant" in the final solution. Schrage has suggested that the flexible tree-search technique introduced by Tuan [1971] can be used to partially alleviate this problem. Given that one has identified some of these "unimportant" variables, our subproblem relaxation can provide some help in this area. If it is suspected that the current assignment of first stage variables cannot lead to an improved integer solution (regardless of what values are given to the currently-unfixed "unimportant" variables), we can suspend the restriction that all first stage variables be assigned values before any second stage variables are assigned. Branching on the second stage variables while leaving the "unimportant" first stage ones free in each subproblem may lead to faster fathoming. Unfortunately, it is not clear just how such "unimportant" variables can be recognized.

Finally, we have insisted that, after fixing all first stage variables, each successive subproblem be solved to optimality before the next one is considered. This can be excessively time-consuming if several subproblems contain some "unimportant" variables, since the various values of these variables would have to be considered before proceeding to the next subproblem. The approach might be improved by allowing partial optimization of successive subproblems in an attempt to verify infeasibility of the first stage assignment or to determine that the assignment cannot lead to an improved objective. If the attempt fails, the method can return to complete the optimization of each subproblem.

CHAPTER 6

COMPUTATIONAL RESULTS

6.1. Computer Code

Evaluation of our subproblem-decomposition algorithm was carried out on an IBM 360/67 computer, using a program written in FORTRAN IV by the author. This program was designed to facilitate experimentation with various alternatives, and therefore the results presented herein should not necessarily be considered representative of those which might be obtained from a professionally-encoded production version. In order to provide a means of comparing the relative effectiveness of our procedure, the code was designed to operate in any of the following three modes:

1. Decomposition to Independent Subproblems -- either of the two versions of our algorithm (sum-of-bounds or surrogate LP) as described in the preceding chapter.
2. Decomposition without First Stage Relaxation -- addressing the original problem formulation with a priority on first stage variables.[30]
3. No Decomposition -- straightforward solution of the problem without exploitation of structure.

Mode 2 is derived from mode 1 by simply not specifying that the values of corresponding unfixed first stage variables are allowed to differ among the subproblem LP solutions. By dropping the requirement that

[30] The decomposition method used by Schrage [1973], and discussed in Section 3.1.

all first stage variables be assigned zero-one values before any such assignments are made to second stage variables, we obtain mode 3 from mode 2.

The linear programming portion of our code was based extensively on a primal-simplex LP code with product-form inverse known as LPM-1, which was written by J.A. Tomlin of the Systems Optimization Laboratory, Department of Operations Research, Stanford University. This program was modified to use the upper bound technique for dealing with unit upper bounds on all variables, and appropriate subroutines were added to permit operation as a dual-simplex method as well. Reoptimization after forward branches was performed with primal-simplex pivots, whereas the dual-simplex method was used upon backtracking when a previously-optimal solution was available. The remainder of our code was written in functional modules according to the various tasks to be performed (branch selection, penalty calculation, testing of bounds, etc.), with a main program to coordinate the application and sequencing of the various modules. The final version of the program was compiled under IBM's FORTRAN-H compiler with the object-program optimization option employed at level two. While dimensioned to handle up to 10 subproblems of 20 rows each with 20 first stage variables and 10 second stage variables per subproblem, the object program required approximately 192 K bytes of main memory (with all instructions and data in-core).

Various options were built into the code to allow monitoring of solution paths, penalties and bounds calculated at each node, methods of branch selection and fathoming, and identification of

improved integer solutions. However, the results presented herein are of runs in which all such intermediate output (except for description of improved solutions) was suppressed. The solution times which we report are based on the total time during which the CPU was available to the core partition containing the job, excluding I/O wait time and the time required to load the object program and data in from disk storage.

6.2. Description of Testing Procedure

The primary objective of the testing program was to determine the solution-time dependence of our algorithms on the number of subproblems and constraint rows in a variety of test problems. This was accomplished by formulating a set of six "master" problems, with each such master composed of at least eight subproblems. By deleting various numbers of subproblems (and their objective function contributions) from each master, additional problems were derived and solved. Test problems obtained from any given master had the advantage of containing many of the characteristics of their source, and were therefore closely related in terms of computational complexity. This relationship made it possible to directly compare the solution times for problems of increasing size, and to attribute the results of such comparisons largely to the influence of size. Had we chosen to compare the solution times of totally unrelated problems of various sizes, it would have been necessary to generate a rather large sample in order to average-out the variations in complexity among problems of equal size.

Limitations on time and funds would necessarily have restricted our investigation to fewer types of problems if this latter alternative had been chosen.

Given a specific master problem and its derivatives, the procedure followed was to solve each member of this set using one or both variants of our decomposition algorithm. Using the solution times thus obtained as a standard, modes 2 and 3 of the preceding section were then applied to successively larger problems from the set until their time requirements grew to be excessive or all problems were solved. As will be shown in the sections dealing with each problem set, mode 1 proved to be clearly superior in all but a few of the smaller problems (with the sum-of-bounds variant outperforming the surrogate LP approach).

The sum-of-bounds variant of our algorithm experienced such a slow growth of solution times with problem size in the first six sets that a rather large test problem (30 subproblems vs. the usual 8-10) was generated and run in various sizes with mode 1. This was done in an attempt to determine if the growth rate would increase significantly as the number of subproblems was enlarged beyond the limits of previous runs. The absence of such acceleration appears to be very promising for the solution of large-scale problems by our subproblem-decomposition algorithm.

The eighth, and final, set of test problems was selected from a group derived by Schrage [1973] from a problem known as IBM-1 (Haldi [1964]). Each of these problems consisted of two subproblems having an equal number of rows and of second stage variables, and differed only in the number of first stage (or linking) variables

retained from the master problem. Even with such a small number of subproblems, our algorithm proved clearly superior to each of the other two methods tried, and showed a more consistent dependence on the number of linking variables than did Schrage's procedure.

6.3. Test Problem Number One

The master of our first test problem set was formulated from the data of an R&D project selection problem (Number 5) devised by Petersen [1967]. The original problem represented a group of 28 projects by a measure of profitability and their budgetary requirements during each of ten years. In order to obtain a problem with the structure for which our algorithm was designed, we first divided each of the original projects into two separate ones of five year's duration each. That is, the first five rows of the constraint matrix were taken to represent one group of 28 revised projects, and the last five rows represented a second group. Each objective function coefficient was divided by 10, rounded up to the nearest even integer, and then split equally among the two projects formed from its column. By alternately listing one project from each group, we obtained a collection of 56 columns of five rows each.

The first six of these columns were chosen to be the first stage matrix entries in each of the subproblems of the dual-angular structure being built. The remaining 50 columns were divided sequentially into 10 groups of 5 columns each. These ten groups

were used as the diagonal submatrices representing the second stage of our test problem. Referring to the notation of Figure 5 in Chapter 4, each of the submatrices A_i $(i = 1, \ldots, 10)$ consisted of the 5 rows and 6 columns of the first group of revised "projects", and each B_i $(i = 1, \ldots, 10)$ was a 5×5 submatrix representing one of the remaining groups of 5 projects each. Each element of the first stage cost vector, c, consisted of the sum of the identical cost rows associated with each A_i $(i = 1, \ldots, 10)$, and each d_i was the second stage cost vector associated with the corresponding B_i $(i = 1, \ldots, 10)$. The right-hand side (RHS) of each of the resulting 50 constraints was determined by taking approximately 70% of the sum of coefficients in each constraint (Petersen seems to have followed the same rule for determining the RHS of his original problems).

The cost and constraint coefficients for each of the 20 submatrices in test problem number one have been tabulated along with the aggregate first stage cost vector, c, in Table A-I in the appendix.

Five different derivative problems, consisting of the first n subproblems (n = 2, 4, 6, 8, 10) of the master, were solved using both versions of the subproblem decomposition algorithm. Attempts were also made to solve the first four of these problems with mode 2 of our code, and the first three were addressed with mode 3. The results of these efforts are shown in Table I and are plotted in Graphs I and II. "Total Nodes Explicitly Considered" refers to those nodes of the enumeration tree whose associated assignment of values to variables was actually made in the course of the solution

Table I

Results of Test Problem Number One

	Algorithm Variant	NUMBER OF SUBPROBLEMS				
		2	4	6	8	10
Total Time in Seconds	DSB	.77	2.02	11.29	29.60	27.84
	DSSC	1.43	3.21	19.38	55.39	60.38
	DFM	1.41	10.03	118.48	> 300	
	NDFM	.96	28.88	148.72		
Total Nodes Explicitly Considered	DSB	14	31	173	464	384
	DSSC	18	33	169	460	384
	DFM	20	59	366		
	NDFM	7	222	715		

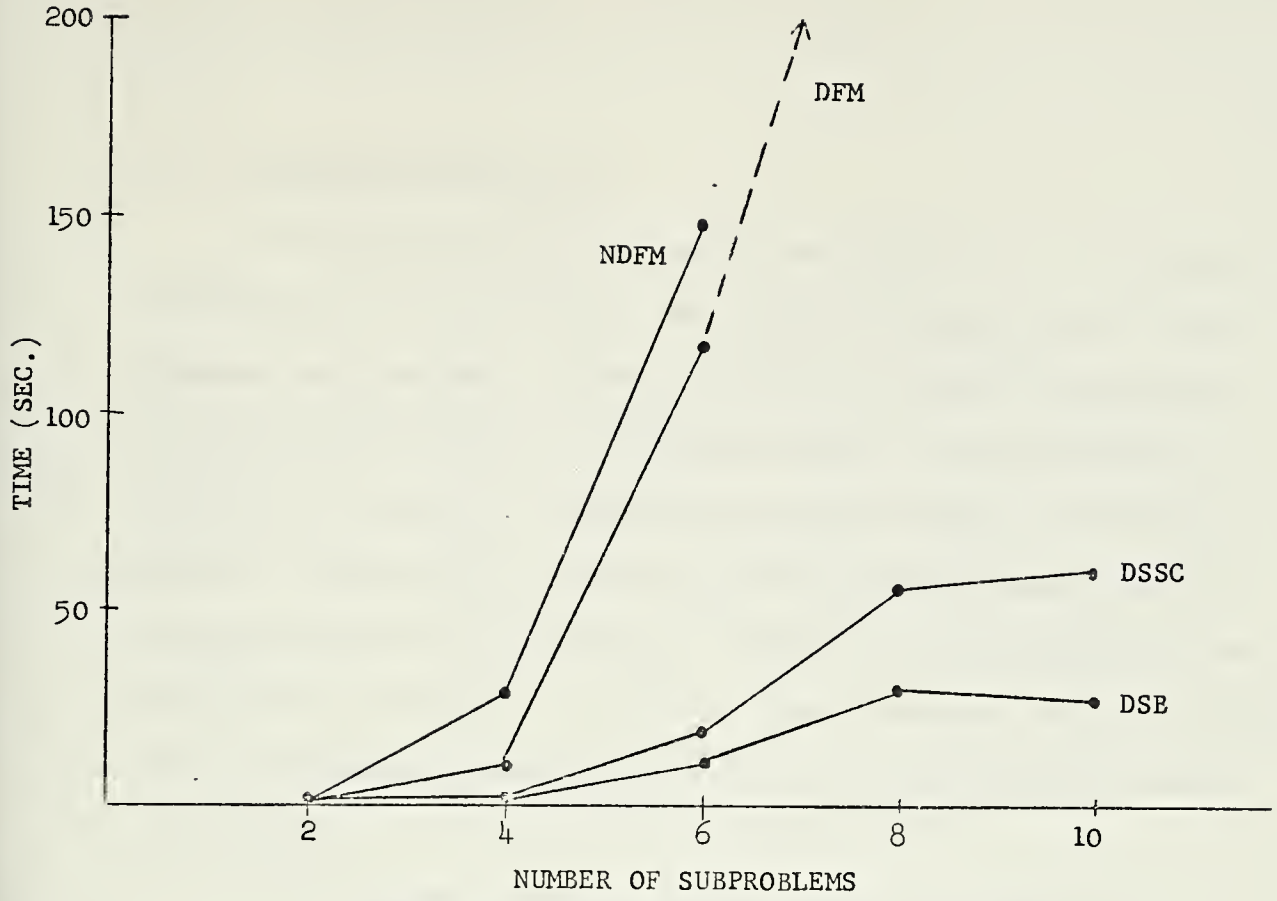
DSB = Mode 1: Sum-of-bounds variant

DSSC = Mode 1: Surrogate LP variant

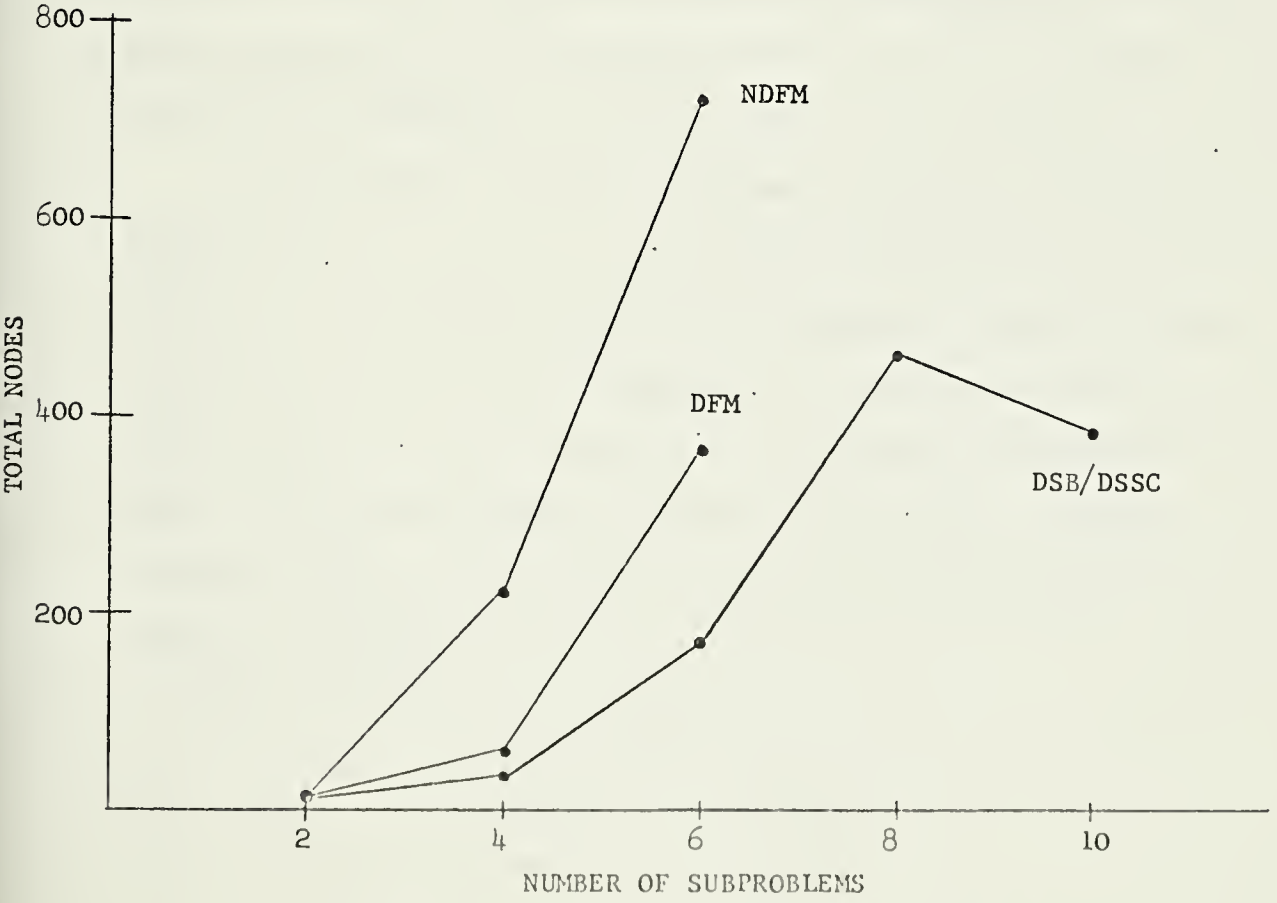
DFM = Mode 2

NDFM = Mode 3

process. Those nodes which were not analyzed explicitly because of predecessor nodes being fathomed are excluded from this count. Each of the algorithms was allowed to take advantage of the fact that all costs were integers by rounding all upper bounds obtained on various branches down to their closest integer values before testing for fathoming.



GRAPH I: Time Results, Test Problem One



GRAPH II: Node Results, Test Problem One

6.4. Test Problem Number Two

The second test problem's master was formulated from the data of Petersen's 5×50 Problem 7 in a manner similar to that of the preceding test problem. Instead of choosing the first six columns as first stage variables, we randomly selected 10 of the 50 columns, discarded the first 4 of these, and used the remaining 6 as linking variables. The other 40 columns were then divided sequentially into 8 groups of 5 and used in 8 diagonal submatrices (second stage). The original objective function coefficients were retained for all variables, and the RHS was once again determined by taking approximately 70% of each row sum.

The ten randomly-selected columns were 1, 7, 9, 14, 19, 31, 35, 38, 40, and 47. The cost and constraint coefficients of each of the 16 submatrices of test problem number two have been tabulated along with the aggregate first stage cost vector in Table A-II.

Four different derivative problems, consisting of the first n subproblems ($n = 2, 4, 6, 8$) of the master were solved by both versions of the subproblem decomposition algorithm. The first three of these were also solved with modes 2 and 3, and an attempt was made to solve the fourth using mode 3. The results are shown in Table II and plotted in Graphs III and IV. Once again, since all costs were integer, rounding was performed on all bounds before attempting fathoming.

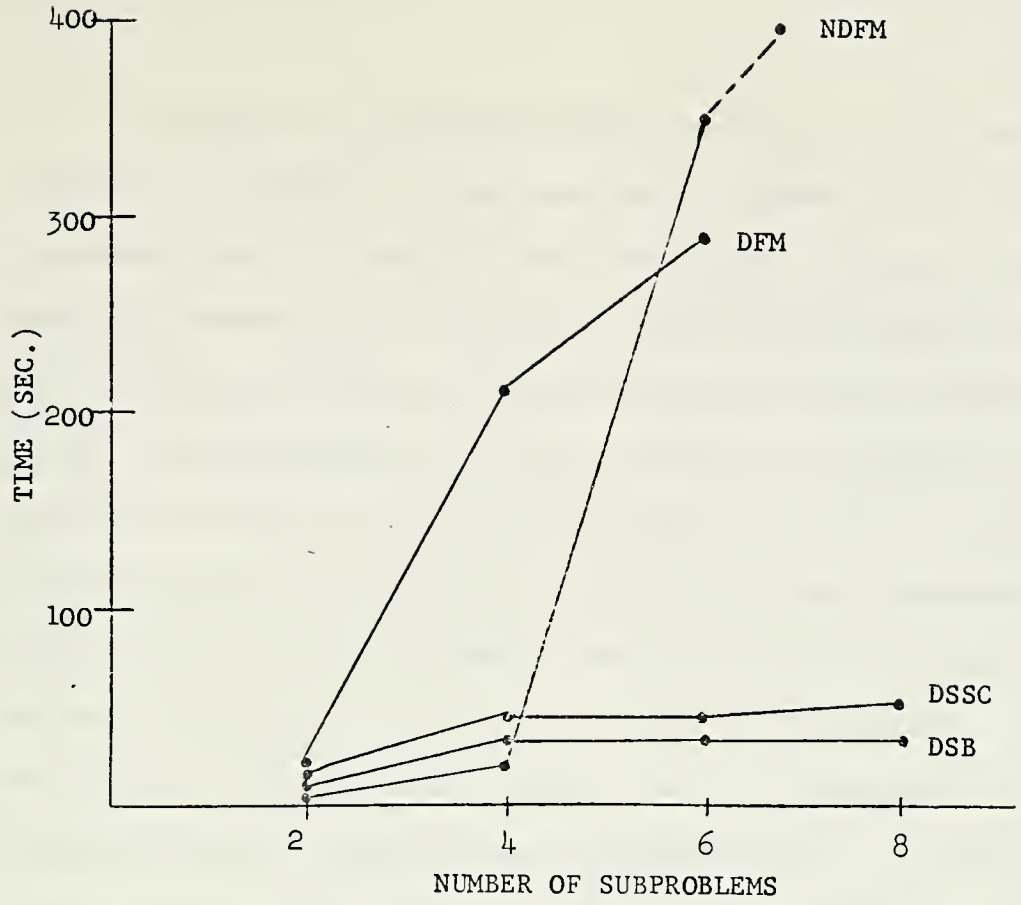
Table II

Results of Test Problem Number Two

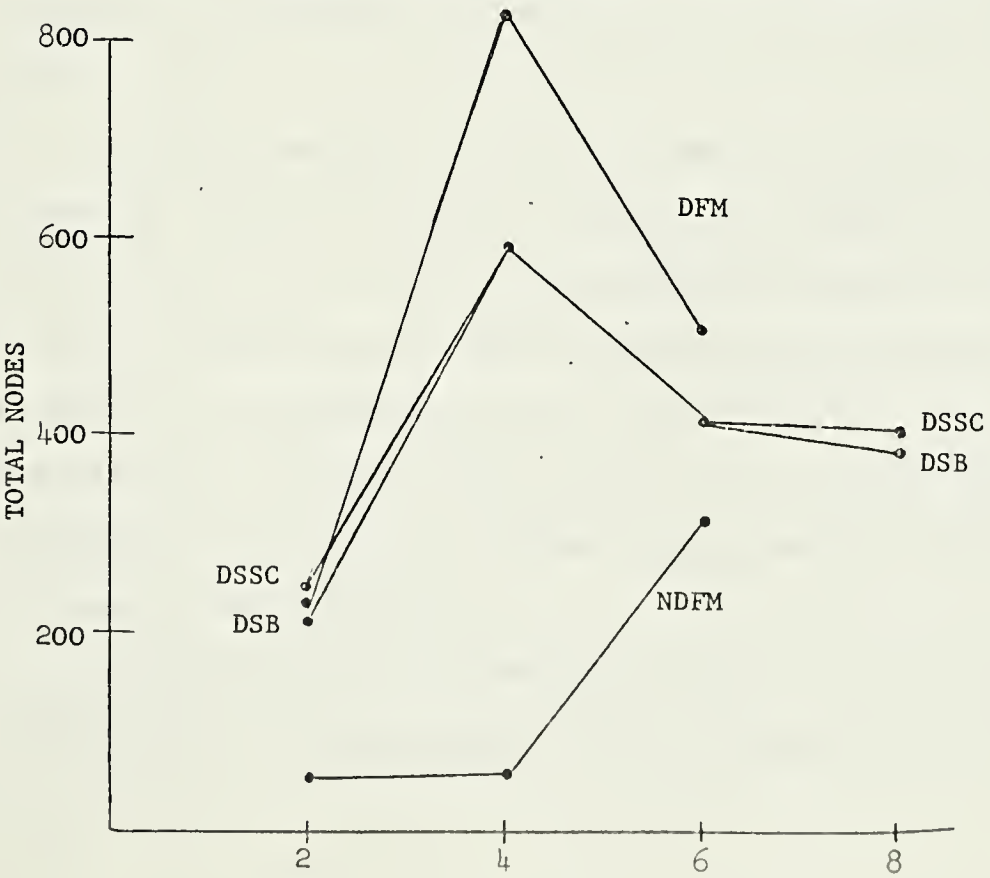
	Algorithm Variant	NUMBER OF SUBPROBLEMS			
		2	4	6	8
Total Time in Seconds	DSB	12.50	35.97	34.07	32.38
	DSSC	16.99	46.86	48.26	54.31
	DFM	24.35	212.20	290.39	
	NDFM	9.36	21.28	352.75	> 480
Total Nodes Explicitly Considered	DSB	216	592	417	382
	DSSC	245	590	422	403
	DFM	224	825	504	
	NDFM	52	57	309	

6.5. Test Problems Three and Four

Since our decomposition algorithm was initially designed to be applied to a class of multi-stage capital budgeting problems, it seemed highly desirable to use some test problems representing this class. Unfortunately, we were unable to find any dual-angular capital budgeting problems in the literature, so a matrix generating program was developed to produce "projects" with random cash flows from specified ranges. The first two such test problems were designed with 4 first stage and 4 second stage variables per subproblem, and the number of possible first stage outcomes (subproblems) was set at 10. These 10 outcomes represented 5 different general states (bad, fair, average, good, excellent) with 2 possible sets of available second stage projects for each such general state.



GRAPH III: Time Results, Test Problem Two



GRAPH IV: Node Results, Test Problem Two

The matrix was generated by first choosing for the "average" case 4 first stage project cash flow sequences as random integers uniformly distributed over the ranges shown in Table III. The first element of each such sequence was designated to be an initial "start-up" cost for its associated project, and was then added to the second element to give total first-year cost for the project. The remaining elements were then used as net cash flows for years 2, 3, and 4 (all projects had a fixed life of four years). The next step was to generate 4 second stage projects in a similar manner, except that they were designated to begin during year 2 and extend through year 5. The second set of possible second stage project availabilities for the "average" case was determined by choosing the first two such projects from the first set and generating two new ones. Therefore, there were two projects which would be available in either case at stage two.

Having accomplished the above, we were in possession of all the constraint coefficients needed to represent the "average" case (sub-matrices A_5/A_6, B_5, and B_6). The remaining four cases were derived from this "average" case by adding randomly-selected increments (or decrements) to each element of the 10 distinct projects, except for the first-year cash flows of first stage projects. These integer increments and decrements were chosen from uniform distributions over the ranges shown in Table III.

Probabilities for occurrence of the five major states were chosen to be 0.1, 0.2, 0.4, 0.2, and 0.1, respectively, from "bad"

Table III

Test Problems Three through Six: Parameters

Time of Flow	Initial Range	Increment Case and Range			
		Bad	Fair	Good	Excellent
Start-up	[-40, -25]				
Year 1	[-25, 0]	[-9, -3]	[-7, 2]	[-2, 7]	[3, 9]
Year 2	[-25, 0]	[-9, -3]	[-7, 2]	[-2, 7]	[3, 9]
Year 3	[30, 50]	[-9, -3]	[-7, 2]	[-2, 7]	[3, 9]
Final Year	[20, 40]	[-9, -3]	[-7, 2]	[-2, 7]	[3, 9]

to "excellent". Within each of these states, probabilities of 0.4 and 0.6 were assigned to the two possible second stage project availabilities, resulting in probabilities of (0.04, 0.06, 0.08, 0.12, 0.16, 0.24, 0.08, 0.12, 0.04, 0.06) for the ten subproblems. Objective function "weights" were determined for the subproblems by dropping the decimals in these probabilities and dividing by their greatest common denominator (2). The subproblem objective function coefficients were then determined by summing each project's cash flows and multiplying by the appropriate objective function "weight".

The final step was to determine a RHS for each constraint in the "average" case by adding 35% of the sum of its positive coefficients to 65% of the sum of its negative ones. The RHS's thus obtained for the two project-availabilities were then averaged and rounded, and this final set of values was used as the budgetary limits (RHS) for every subproblem. All constraints were "greater-than-or-equal-to" inequalities.

The cost and constraint coefficients of test problems three and four are listed in Tables A-III and A-IV.

Each of these two problems was run in five sizes formed by successively removing the subproblems corresponding to the bad, excellent, fair, and good outcomes, in that order. All 10 problems were solved with modes 1 (sum-of-bounds variant), 2, and 3 of our algorithm. The results of these efforts are presented in Tables IV and V and in Graphs V - VIII. Rounding of all bounds down to nearest integers was employed throughout.

Table IV

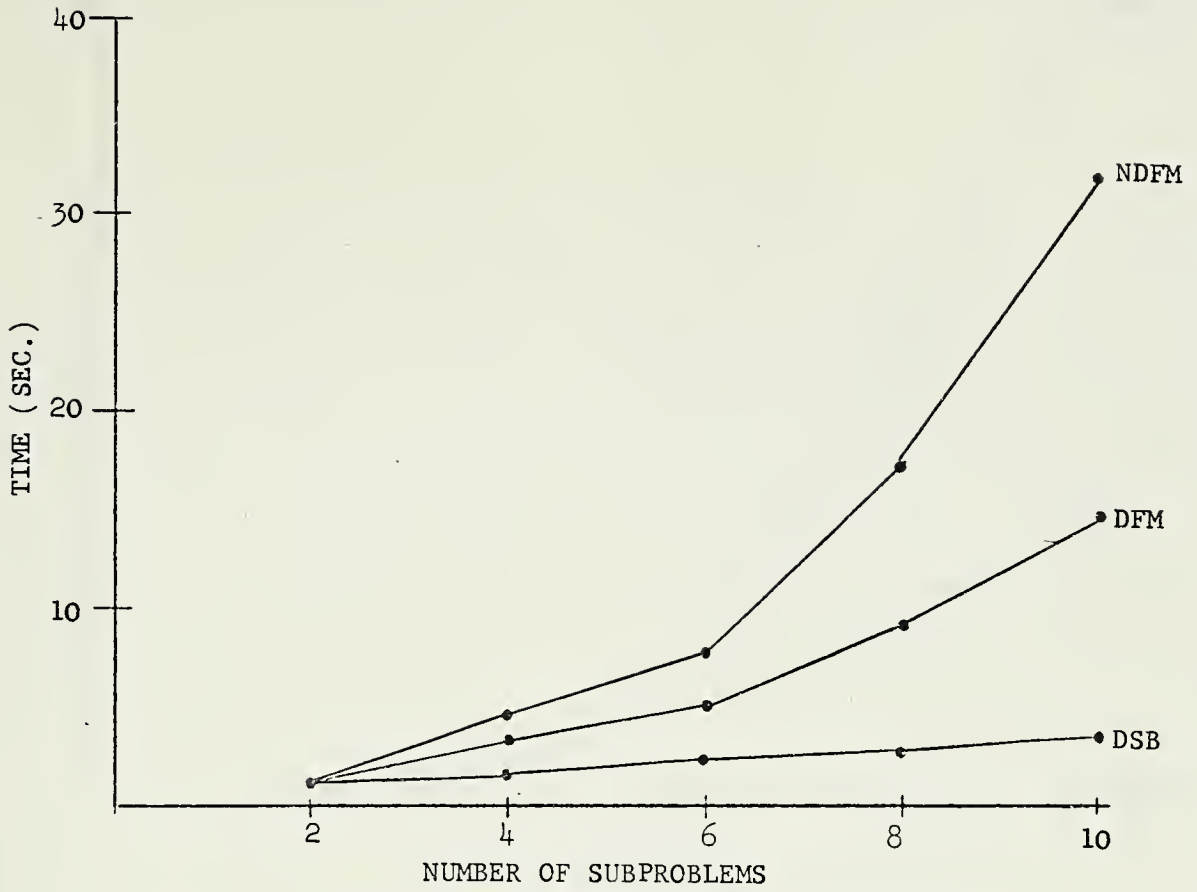
Results of Test Problem Number Three

	Algorithm Variant	NUMBER OF SUBPROBLEMS				
		2	4	6	8	10
Total Time in Seconds	DSB	1.03	1.64	2.38	2.95	3.59
	DFM	1.17	3.27	5.09	9.09	14.81
	NDFM	1.32	4.71	7.76	17.18	31.85
Total Nodes Explicitly Considered	DSB	21	25	28	35	37
	DFM	22	28	28	36	40
	NDFM	29	55	55	94	135

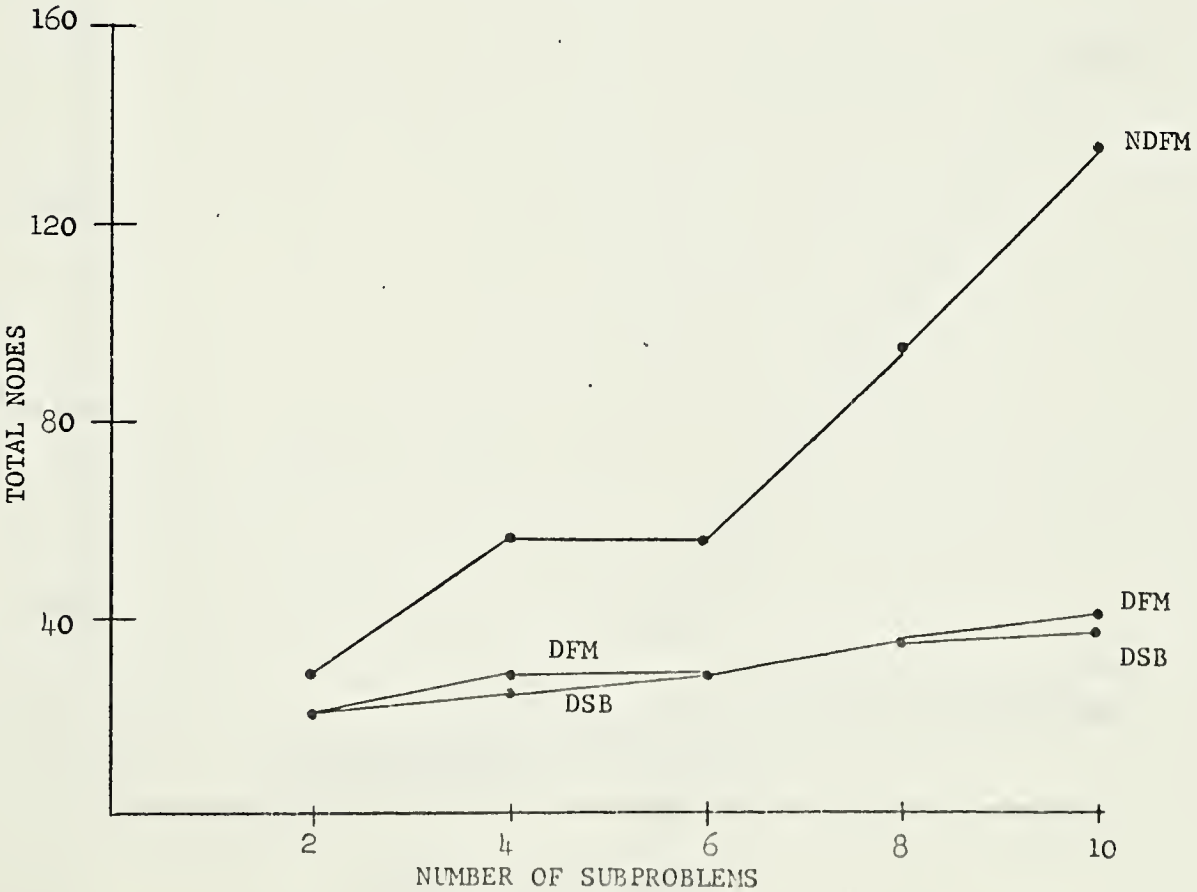
Table V

Results of Test Problem Number Four

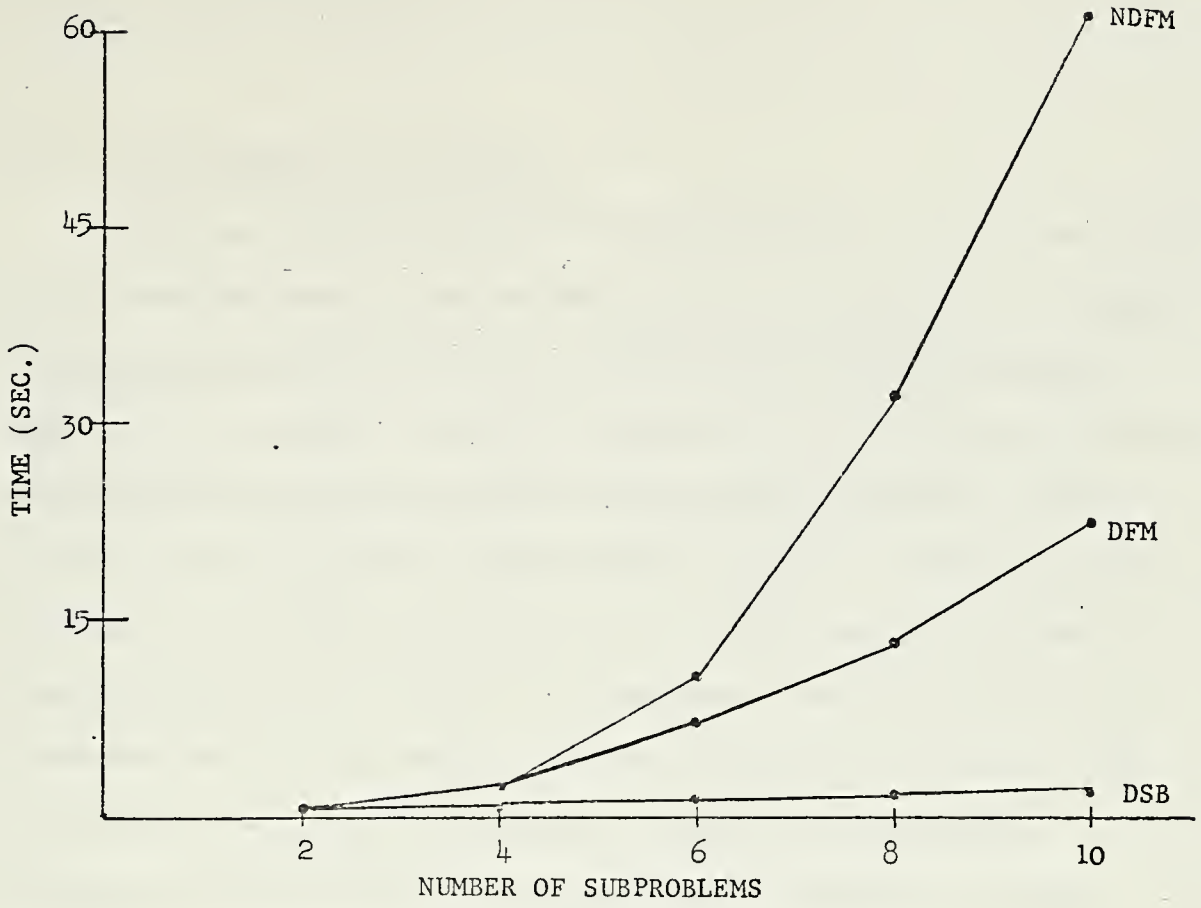
	Algorithm Variant	NUMBER OF SUBPROBLEMS				
		2	4	6	8	10
Total Time in Seconds	DSB	.34	.80	1.19	1.61	2.09
	DFM	.72	2.64	7.57	13.20	22.25
	NDFM	.57	2.76	10.60	32.68	61.95
Total Nodes Explicitly Considered	DSB	9	16	22	29	38
	DFM	12	21	31	39	49
	NDFM	8	24	69	163	269



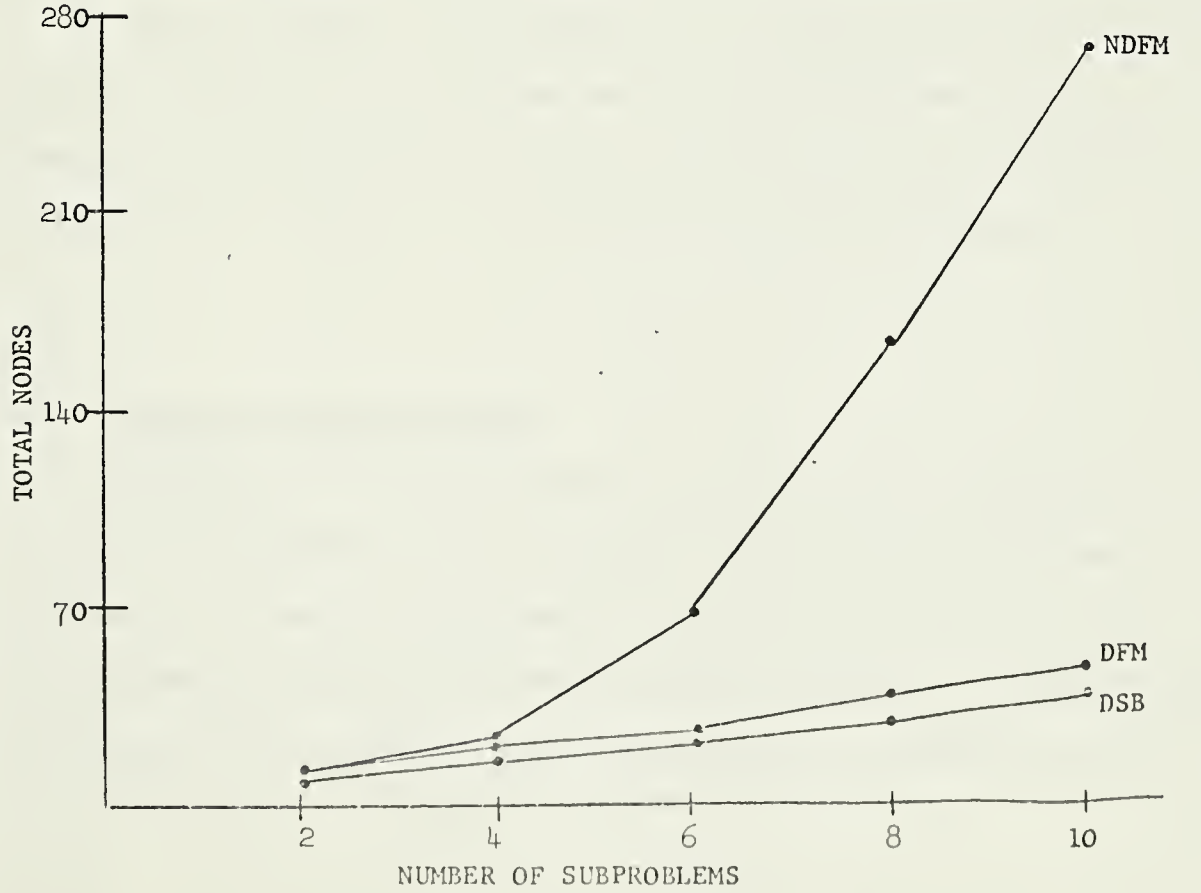
GRAPH V: Time Results, Test Problem Three



GRAPH VI: Node Results, Test Problem Three



GRAPH VII: Time Results, Test Problem Four



GRAPH VIII: Node Results, Test Problem Four

6.6. Test Problems Five and Six

The very encouraging results obtained by our subproblem decomposition algorithm (DSB) in problems three and four, especially the somewhat less-than-proportional increase of solution times with the number of subproblems attempted, led to the formulation of two more such problems. The main difference in these additional problems was that they were designed with 5 projects in each stage instead of 4. The same matrix generator and parameters (Table III) were used along with the same probabilities for each of the 10 subproblems. The only difference in procedure was that the RHS was calculated using 45% and 75% (vs. 35% and 65%) of the positive and negative constraint coefficients, respectively. The objective and constraint coefficients of these problems are presented in Tables A-V and A-VI.

These additional test problems were run using the same algorithm variants applied to the preceding two, and the results are given in Tables VI and VII and Graphs IX - XII. Integer objective coefficients again allowed all bounds to be rounded down to nearest integers.

6.7. Test Problem Number Seven

Experience with the test problems discussed above (especially problems 3-6) indicated a marked tendency for the solution times and node counts to increase approximately linearly with problem size when the subproblem-decomposition algorithm (DSB) was applied. In retrospect, this result is not particularly surprising since the algorithm largely

Table VI

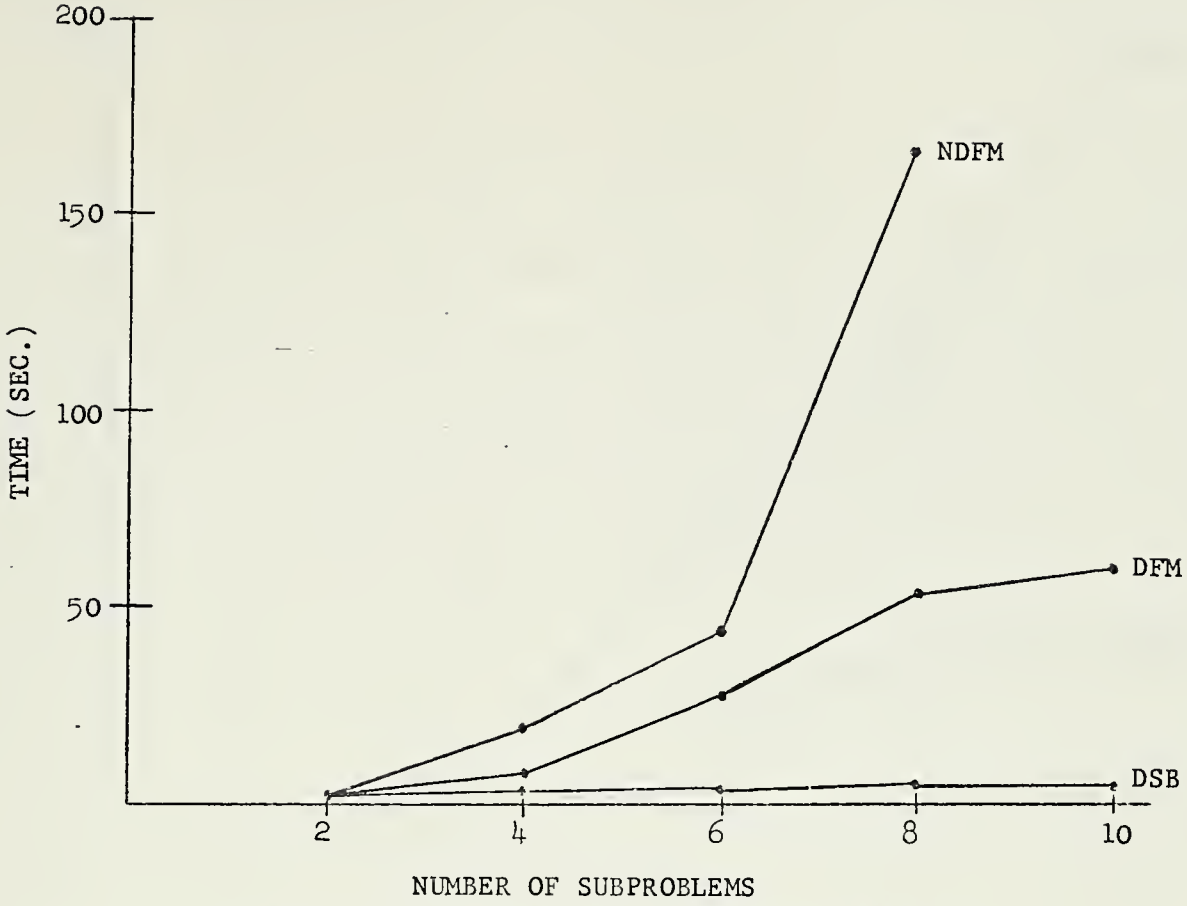
Results of Test Problem Number Five

	Algorithm Variant	NUMBER OF SUBPROBLEMS				
		2	4	6	8	10
Total Time in Seconds	DSB	1.38	2.39	3.34	5.06	5.00
	DFM	1.84	7.07	28.06	54.76	60.65
	NDFM	1.79	19.29	44.08	166.30	--
Total Nodes Explicitly Considered	DSB	29	31	34	47	45
	DFM	25	51	139	183	128
	NDFM	26	194	288	818	--

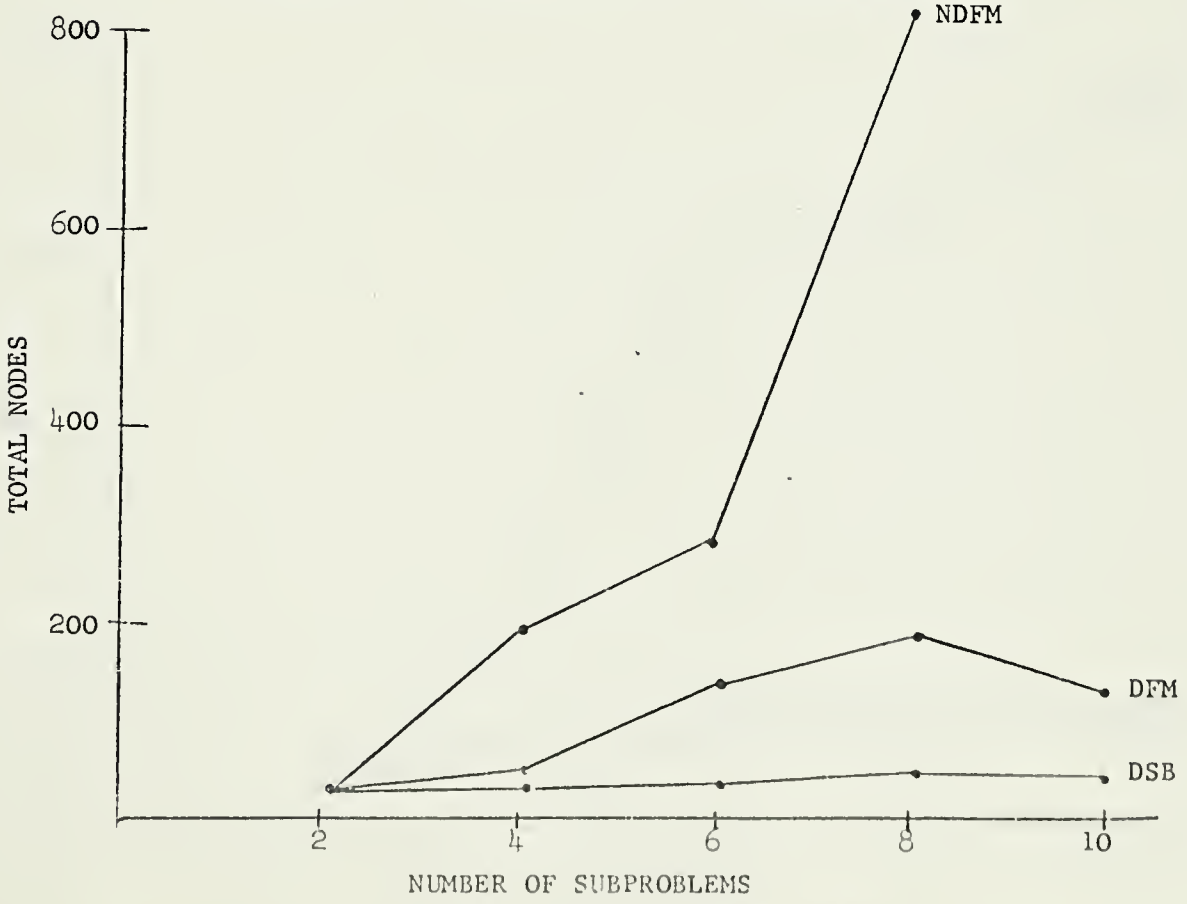
Table VII

Results of Test Problem Number Six

	Algorithm Variant	NUMBER OF SUBPROBLEMS				
		2	4	6	8	10
Total Time in Seconds	DSB	.62	1.31	2.10	2.32	2.71
	DFM	1.10	3.47	10.22	15.20	18.35
	NDFM	.87	6.68	30.69	45.04	77.00
Total Nodes Explicitly Considered	DSB	16	26	34	34	37
	DFM	19	30	39	39	43
	NDFM	13	73	224	225	273



GRAPH IX: Time Results, Test Problem Five



GRAPH X: Node Results, Test Problem Five



GRAPH XI: Time Results, Test Problem Six



GRAPH XII: Node Results, Test Problem Six

ignores the interrelationships among subproblems and treats each additional one as a separate integer program to be solved in parallel with the rest. In effect, doubling the total size of a problem requires not the periodic solution of a linear program which is twice as large, but the periodic solution of twice as many linear programs of the same size. The same relationship holds true with respect to calculation of penalties and performance of the various bookkeeping and overhead operations. The fact that the last four problems gave rise to more consistent results than the first two may be attributed to the formers' similarity among subproblems, as was noted in Section 2 of this chapter. Such similarities would be expected in "real-world" multi-stage capital budgeting problems, which our matrix generator was designed to simulate.

In order to obtain additional information on the relationship between problem size and computational effort with our algorithm, the matrix generator was used to produce a large problem (30 subproblems) with characteristics identical to those of test problems 3 and 4. Each subproblem consisted of 6 rows (including the objective), 4 first-stage variables, and 4 second stage variables. The problem was run in 8 different sizes using the subproblem-decomposition method (DSB), and the results are given in Table VIII and Graphs XIII and XIV.

The outcome of this additional test series obviously reinforces our contention that the subproblem-decomposition algorithm will generally experience a linear growth of solution time with problem size when applied to dual-angular structures of the type herein considered, especially if there exists a similarity or consistency

Table VIII

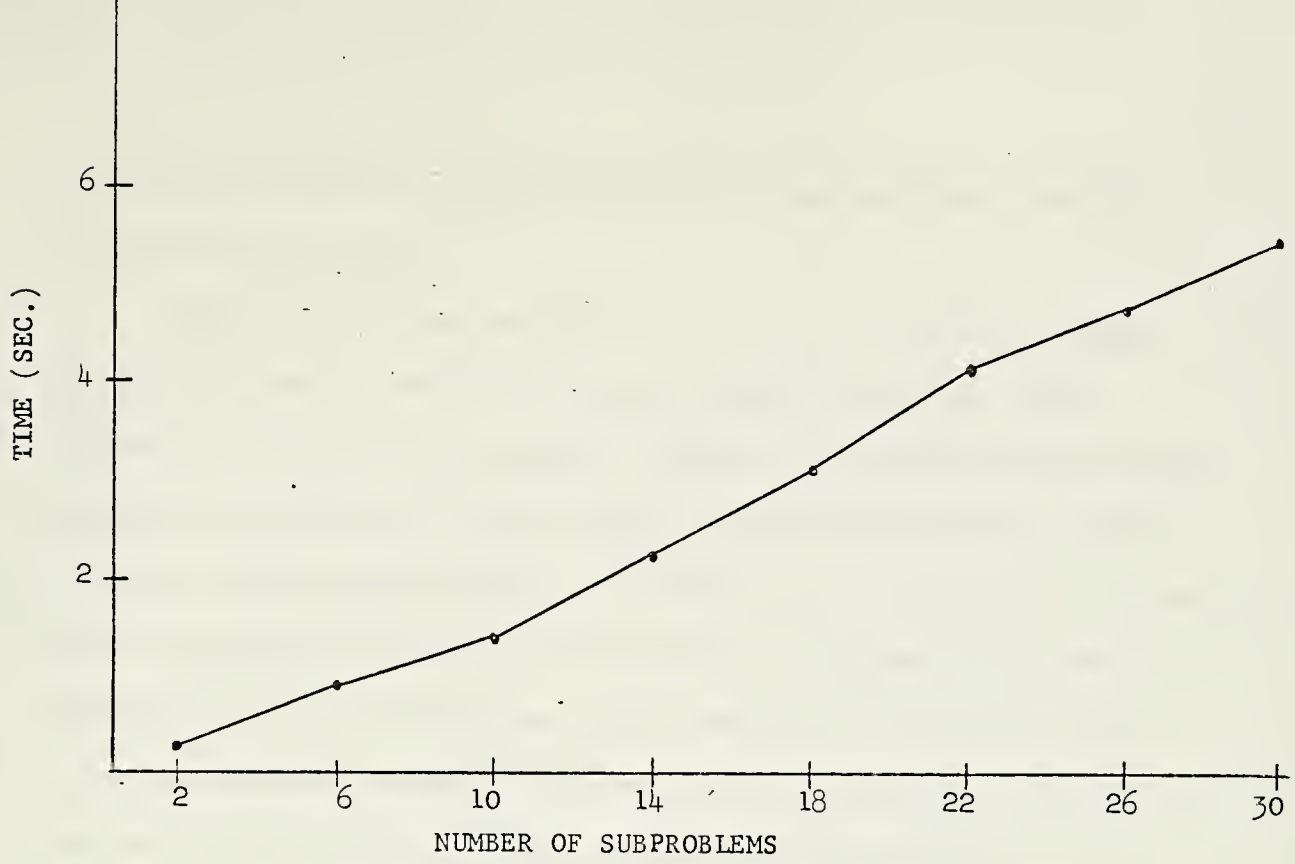
Results of Test Problem Number Seven

	NUMBER OF SUBPROBLEMS							
	2	6	10	14	18	22	26	30
Total Time in Seconds	.34	.91	1.40	2.24	3.10	4.18	4.79	5.51
Total Nodes Explicitly Considered	8	12	15	24	33	46	50	54

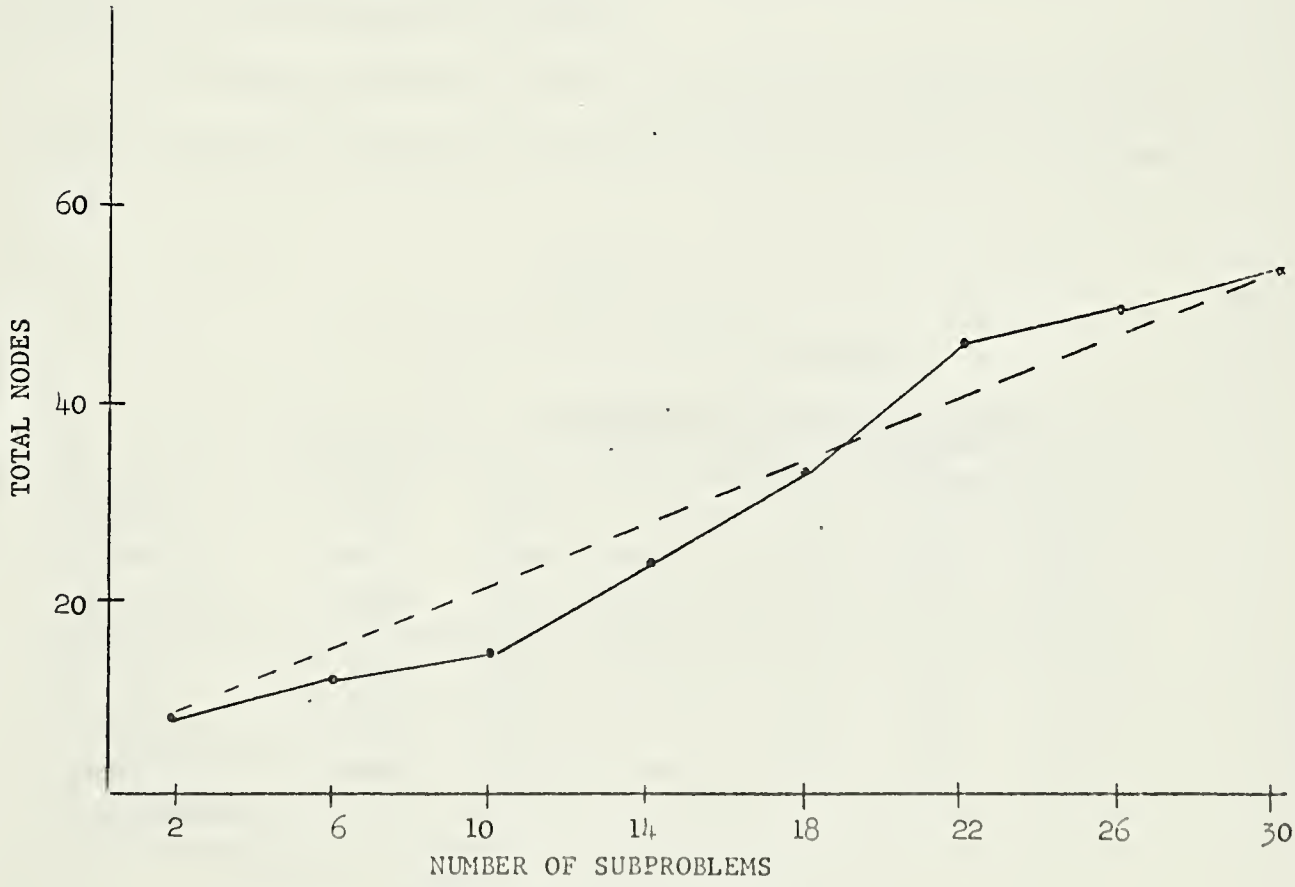
among subproblems. Additional testing of this hypothesis on a wider range of problem types would of course be desirable, but indications are that the method will provide a viable technique for the solution of large-scale dual-angular integer programs.

6.8. Schrage's Test Problem

When applied to the two-subproblem versions of our first six test problems, the subproblem-decomposition algorithm obtained results quite similar (and sometimes inferior) to those obtained by modes 2 and 3 of the code. Despite the fact that the algorithm's relative efficiency could be questioned at this lower level, it was decided to apply the method to a selection of 5 of the 8 test problems devised by Schrage [1973]. The main purpose of this investigation was to gain insight into the behavior of the three algorithm modes as we increased the number of linking variables (first stage) while holding



GRAPH XIII: Time Results, Test Problem Seven



GRAPH XIV: Node Results, Test Problem Seven

the number of subproblems (two) and second stage variables (fourteen per subproblem) constant.

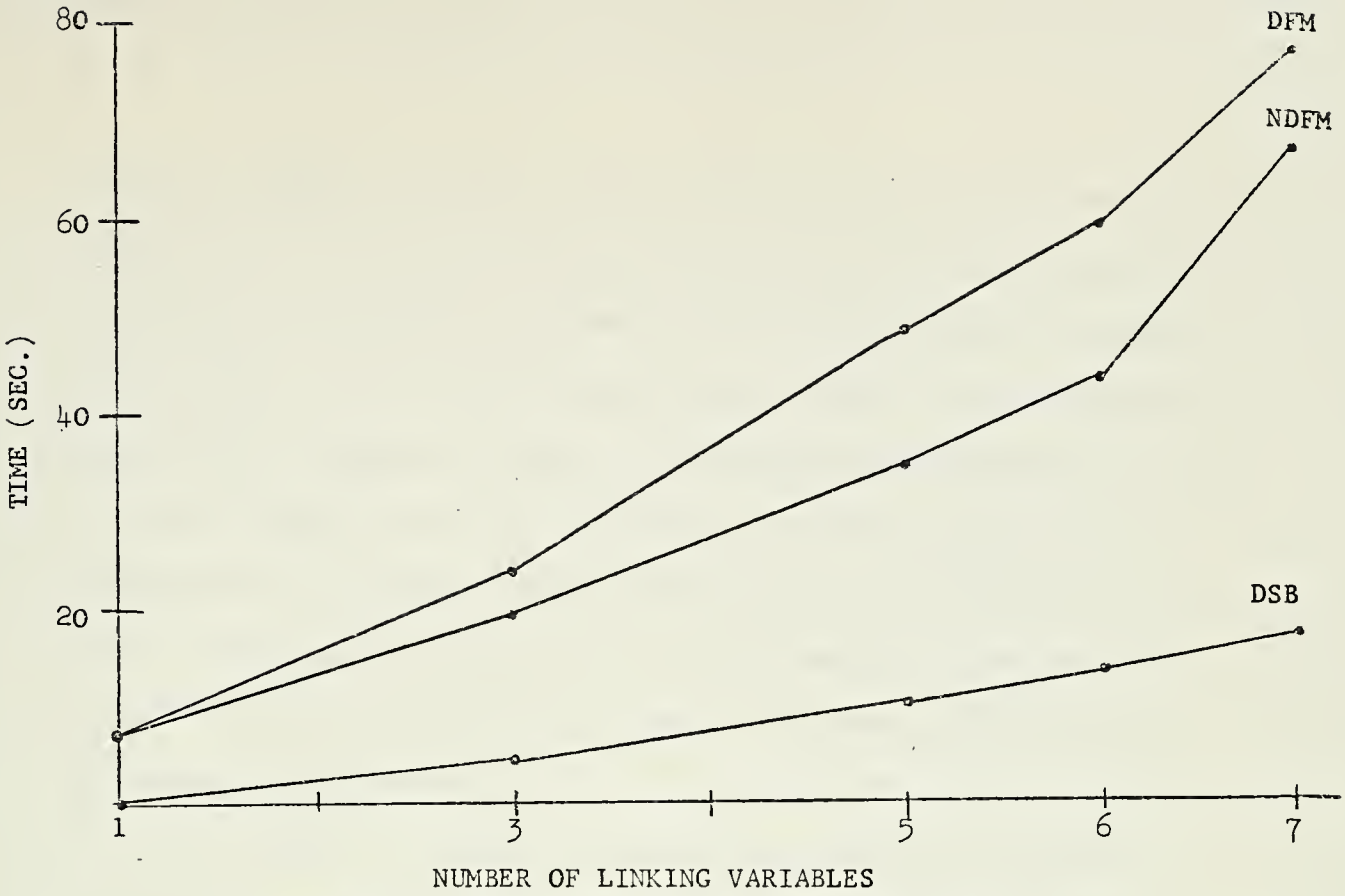
Since there existed no natural allocation of each first stage variable's objective coefficient among the two subproblems, equal division was arbitrarily specified. Rather than doubling the resulting subproblem objectives to obtain integer costs, we chose not to enable the bound-rounding feature used in previous runs. In addition to other tactical differences between our algorithm and Schrage's, this lack of rounding in our runs renders any direct comparison of solution times or node counts with Schrage's reported results meaningless. We therefore base our conclusions solely on the relative performances of our algorithm's three modes, and draw no conclusions concerning the efficiency of our method vs. Schrage's.

The results reported in Table IX and shown in Graphs XV and XVI indicate that the subproblem-decomposition algorithm again outperformed

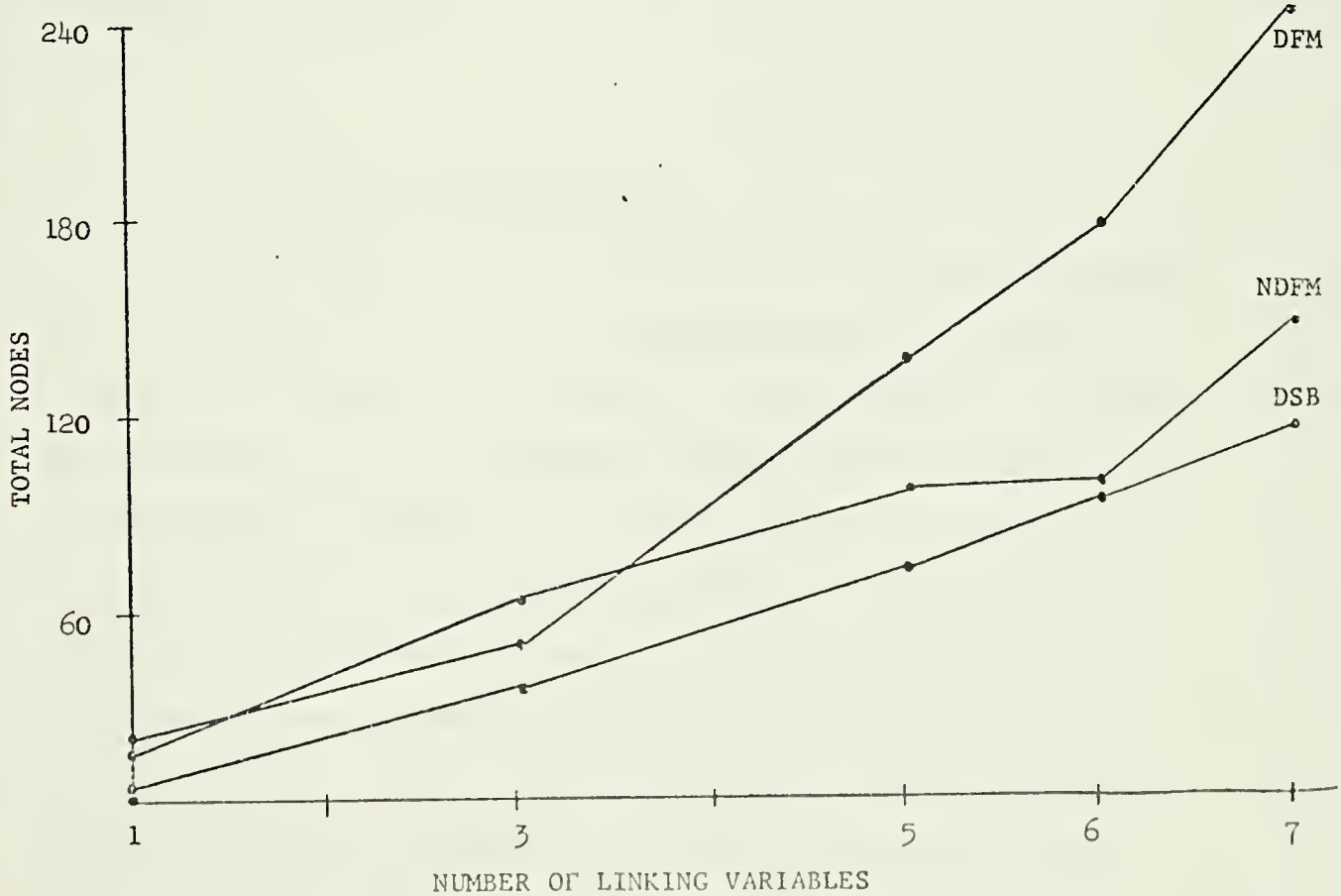
Table IX

Results of Schrage's Test Problem

	Algorithm Variant	NUMBER OF LINKING VARIABLES				
		1	3	5	6	7
Total Time in Seconds	DSB	1.09	4.51	9.97	13.41	16.90
	DFM	7.80	23.43	47.90	58.60	76.87
	NDFM	6.97	19.63	34.02	43.04	66.63
Total Nodes Explicitly Considered	DSB	7	34	70	92	114
	DFM	21	48	135	176	243
	NDFM	17	61	95	96	145



GRAPH XV: Time Results, Schrage's Test Problem Without Rounding



GRAPH XVI: Node Results, Schrage's Test Problem Without Rounding

both other methods, but with a fairly constant ratio of solution times. The increase of solution times with the number of linking variables is approximately linear with each method, particularly that of subproblem-decomposition. It is also interesting to note that the method without decomposition (NDFM) had a higher node count than the straight decomposition method (DFM) in only one instance, and that its solution times were consistently better. This might be construed as indicating that Schrage's problem(which has only two subproblems) is not particularly amenable to such decomposition, and might therefore explain his somewhat mixed results with it.

The results of this test series give no reason to conclude that decomposition loses its effectiveness as the number of linking variables is increased.

6.9. Summary of Computational Results

Results obtained from the previously described testing procedure strongly indicate that the subproblem-decomposition algorithm enjoys significant advantages over the other two methods tested. The solution times experienced with this method are quite commonly 50 - 80% below those obtained from mode 2 (DFM), which also relies on decomposition via priority specification of linking variables but performs all its operations on the full problem matrix. In the majority of cases with our first six test problems, both of these algorithms have provided much better results than a straightforward method taking no advantage of structure (NDFM). Comparison of the node counts obtained with

DSB/DSSC vs. DFM indicate that, rather than causing decreases in fathoming ability, the separation of subproblems actually may result in an improved capability to recognize unpromising first and second stage branches. It is believed that the obvious weakening of objective bounds which occurs when subproblems are uncoupled is often compensated by the advantage of using each subproblem's largest Gomory penalty rather than a single such penalty taken from the full linear programming tableau. The advantage of having strong bounds for each subproblem seems to be particularly pronounced after a problem has decomposed naturally and integer optima are sought for the independent subproblems. Reference to steps 13 and 16 of our algorithm (Chapter 5, Section 5.4) will clarify the reasons for this claim.

The surrogate-LP version of our algorithm consistently required more time and considered more nodes than its companion sum-of-bounds approach. A separate timer was placed on that portion of the code which set up, optimized, and calculated penalties for the surrogate LP. The time required for these efforts accounted for approximately 50-85% of the additional solution times for DSSC vs. DSB. The remaining time can be attributed to consideration of additional nodes. As was discussed in Sections 5.3 and 5.5, there exist several possible methods for reducing the time required by our surrogate LP option and enhancing its fathoming capabilities; however, it is doubtful that such procedures can give this option a significant advantage over the sum-of-bounds approach.

Our algorithm's approximately linear growth of solution times and node counts with problem size (as determined by the number of

subproblems considered) make it an extremely promising technique for the solution of large-scale dual-angular integer linear programs.

CHAPTER 7

CONCLUSIONS AND AREAS FOR FUTURE RESEARCH

7.1. Capital Budgeting Formulation

As presented in Chapter 3, our extension of Hillier's basic two-stage model to allow consideration of uncertain future availability of projects adds another dimension of realism to a promising theoretical technique for the solution of multi-period capital budgeting problems. In the past, many mathematical programming approaches to capital budgeting have relied on an assumption that all present and future investment opportunities (within the time frame being considered) could be identified and incorporated in the analysis. The applicability of such techniques to "real world" situations has been limited by that assumption because, in most situations, very few possible future events can be regarded as being "certain" to occur. In an environment of uncertainty, one can impose constraints on future cash positions to provide a source of funds for possible investment in desirable opportunities which might arise. This type of approach is rather artificial because it requires that the amounts to be kept available be determined outside of the solution process for the model, thereby reducing or eliminating explicit analytical use of whatever partial information on future events may be available.

At the price of expanding Hillier's model and increasing its computational complexity, we are able to incorporate some of this partial

information directly in the analysis. This is accomplished by relaxing the aforementioned assumption to require predictions of which projects may become available and estimations of their associated probabilities. In practice, of course, consideration would be given only to those projects, or combinations of projects deemed to have the most influence on a model's behavior.

Requirements for predictions of possible investment opportunities and estimations of probabilities demand additional management input to an analysis and more thorough understanding by analysts of the uncertain environment which they are attempting to idealize. Philosophically speaking, this is a very desirable secondary benefit of utilizing the proposed approach. From a more practical point of view, solution of a large model with explicit consideration of future uncertainties should produce much more useful results than would the independent processing of a large number of small problems based on rigid assumptions about the future. Given that the latter approach is taken, one is left with the formidable task of resolving a single "best" course of immediate action from a set of indications which may differ considerably because of their underlying assumptions. This problem is even more difficult in a capital budgeting context, wherein many or all decisions are of the "accept-reject" variety, and therefore not amenable to compromise.

Having argued the case for a more refined and computationally-complex analysis, it is necessary to address the problem of solving the resulting formulation. We believe that the subproblem-decomposition method presented in Chapters 4 through 6 will prove to be a very

efficient technique for the solution of that class of integer programs which includes our formulation of multi-period capital budgeting with uncertain future project availabilities.

7.2. Subproblem-Decomposition Method

The computational experience reported in Chapter 6 strongly indicates that our technique of "uncoupling" column-linked subproblems and then using an enumeration procedure to gradually retrieve the original problem structure is quite efficient for the solution of dual-angular integer programs. Our specific implementation of the method within the framework of a penalty-type algorithm resulted in substantial improvements over both of the other previously-developed techniques which were tested. These improvements were noted not only in total solution times, but also in the numbers of nodes explicitly considered by the various methods. Of equal, or perhaps greater, importance is the approximately linear growth of solution times with the number of subproblems that was experienced with the subproblem-decomposition algorithm. Assuming that this observed behavior is indeed characteristic of the relaxation procedure, the computational tractability of large-scale dual-angular integer linear programs will be determined mainly by that of their component subproblems.

7.3. Areas for Future Research

The field of capital budgeting contains many unresolved problems for both financial experts and operations researchers. Development of procedures for determining the complex interrelationships among investment proposals and incorporating these into valid mathematical models, especially when non-linearities and discontinuities are involved, merits additional study. The same is true of methods for incorporating attitudes toward risk into mathematical idealizations of the decision process; present practices generally fall into two categories -- those which are acknowledged to be somewhat artificial but relatively easy to implement (e.g., chance-constrained programming), and those which have a sound theoretical foundation but cannot be readily obtained or employed (e.g., utility functions). Also, the difficulty of determining a firm's exact "cost of capital" for use in analysis may be at least partially avoided by developments in the area of post-optimality sensitivity analysis for integer programming.

We have already given considerable attention in Chapters 4 and 5 to possible alternatives and modifications in our subproblem-decomposition method. As mentioned previously, the method can be applied within the frameworks of most popular implicit enumeration algorithms, and it would be desirable to obtain indications of its effectiveness in a number of different procedures. In particular, the surrogate constraint algorithm developed by Geoffrion [1969] appears to be quite efficient and would be a prime candidate for synthesis with our method. Additional development of the use of subproblem

surrogate constraints and surrogate linear programs might well prove to be a fertile area for research. It should also be noted that, although we have restricted our discussions to problems involving binary variables, the method can be extended in a direct way to deal with general integer variables.

The majority of test problems used in this study have had equal, or nearly-equal, numbers of first and second stage variables per subproblem. Although such a situation would be common with capital budgeting problems, other types of models (such as those in which stages do not correspond to successive time periods) may not possess such a structure. For this reason, it would be interesting to consider a broader range of test problems of two and more stages. In particular, the behavior of our method in two-stage problems when the number of second stage variables is gradually reduced to zero, if it remains successful, would indicate that problems with many rows might be solved more efficiently if first relaxed to a number of smaller "subproblems" and then reunited by the enumeration process. The question of how to assign subproblem objective coefficients in such a situation (and others in which no natural allocation exists) also deserves serious attention.

In Chapter 3, we discussed four cases in which continuous decision variables appearing prior to the completion of a problem's decomposition could be handled with little difficulty. We also pointed out that continuous variables which appear only in the final level of decomposition create no problems for the decomposition

approach. Obviously, the development of techniques for employing decomposition in problems with general continuous variables not covered by these cases would be of tremendous value for solving dual-angular mixed integer programs.

Finally, for other types of structured integer programs, it is hoped that the success of our technique will kindle greater interest in pre-solution relaxations which are progressively retightened as the enumeration process evolves. We suggest the block-angular structure with linking rows as one to which such a technique might be applied.

APPENDIX

Table A-I

Test Problem Number One

A_i		COLUMNS 1	2	3	4	5	6
	c_i	5	5	11	11	5	5
	1	8	5	24	11	13	7
R	2	8	0	44	0	13	1
O	3	3	3	6	4	4	5
W	4	5	3	9	6	6	9
	5	5	3	11	8	7	9

$(i = 1, \ldots, 10)$

$c = (50, 50, 110, 110, 50, 50)$

B_1		COLUMN 1	2	3	4	5	RHS
	d_1	20	20	15	15	20	
	1	80	55	70	40	80	275
R	2	100	10	100	4	90	260
O	3	20	20	20	14	30	90
W	4	40	30	30	29	40	145
	5	50	35	40	29	40	165

B_2		COLUMN 1	2	3	4	5	RHS
	d_2	20	10	10	6	6	
	1	40	45	21	15	9	195
R	2	10	75	0	25	6	200
O	3	20	8	6	3	12	65
W	4	20	16	12	5	12	100
	5	20	19	16	7	15	120

B_2		COLUMN 1	2	3	4	5	RHS
	d_3	8	8	29	29	20	
R	1	28	18	90	29	130	310
O	2	28	0	120	6	130	315
W	3	12	10	14	18	40	95
	4	18	10	24	30	60	155
	5	18	10	29	30	70	175

B_4		COLUMN 1	2	3	4	4	RHS
	d_4	20	7	7	5	5	
R	1	70	32	21	20	17	215
O	2	32	32	3	40	0	190
W	3	42	6	9	3	12	80
	4	42	16	18	11	18	130
	5	42	21	20	17	18	150

B_5		COLUMN					
		1	2	3	4	5	RHS
	d_5	65	65	33	33	16	
	1	120	35	40	25	30	280
R	2	160	70	40	10	60	355
O	3	20	100	5	20	0	130
W	4	30	110	25	20	10	190
	5	30	120	25	20	15	210

B_6		COLUMN					
		1	2	3	4	5	RHS
	d_6	16	24	24	4	4	
	1	20	20	25	6	5	160
R	2	0	55	0	10	0	160
O	3	5	5	6	3	4	48
W	4	15	13	18	5	7	95
	5	20	25	22	5	7	120

B_7		COLUMN					
		1	2	3	4	5	RHS
	d_7	3	3	130	130	155	
	1	3	2	180	110	220	465
R	2	6	0	240	0	290	490
O	3	0	1	20	20	30	80
W	4	1	2	80	40	60	180
	5	1	3	100	50	70	220

B_8		COLUMN					
		1	2	3	4	5	RHS
	d_8	155	55	55	48	48	
	1	70	50	55	30	20	260
R	2	30	80	10	90	0	265
O	3	50	40	30	10	5	90
W	4	60	50	50	20	25	200
	5	60	55	55	20	25	215

B_9		COLUMN					
		1	2	3	4	5	RHS
	d_9	23	23	15	15	11	
	1	50	50	12	20	5	200
R	2	70	10	27	10	17	210
O	3	0	20	5	20	0	65
W	4	30	25	10	25	5	120
	5	50	30	15	25	15	160

B_{10}		COLUMN					
		1	2	3	4	5	RHS
	d_{10}	11	10	10	26	26	
	1	15	8	6	18	20	150
R	2	5	8	0	28	10	310
O	3	10	0	10	10	20	65
W	4	15	3	10	20	28	110
	5	15	6	10	20	28	120

Table A-II

Test Problem Number Two

A_i	COLUMN 1	2	3	4	5	6
c_i	82	49	71	116	738	620
1	1	0	4	10	40	28
2	2	4	3	0	28	45
3	3	5	8	0	6	14
4	0	2	1	20	14	20
5	6	0	3	5	0	22

$(i = 1,\ldots,8)$

$c = (656, 392, 586, 928, 5904, 4960)$

B_1	COLUMN 1	2	3	4	5	RHS
d_1	1125	300	620	2100	431	
1	91	10	30	160	20	275
R 2	92	41	16	150	23	280
O 3	39	32	71	80	26	205
W 4	71	30	60	200	18	300
5	52	30	42	170	9	235

B_2	COLUMN 1	2	3	4	5	RHS
d_2	328	122	322	196	41	
1	12	18	9	25	1	100
R 2	18	0	12	8	2	85
O 3	40	12	30	15	0	100
W 4	30	8	31	6	3	90
5	20	3	21	4	1	55

B_3		COLUMN					
		1	2	3	4	5	RHS
	d_3	425	4260	416	115	22	
	1	10	280	10	8	1	275
R	2	0	200	20	6	1	315
O	3	23	100	0	20	0	130
W	4	18	60	21	4	2	110
	5	14	310	8	4	1	260

B_4		COLUMN					
		1	2	3	4	5	RHS
	d_4	631	132	420	86	42	
	1	49	8	21	6	1	115
R	2	70	9	22	4	1	130
O	3	40	6	8	0	6	75
W	4	32	15	31	2	2	95
	5	18	15	38	10	4	85

B_5		COLUMN					
		1	2	3	4	5	RHS
	d_5	103	215	81	91	26	
	1	5	10	8	2	1	75
R	2	5	10	6	4	0	75
O	3	4	22	4	6	1	55
W	4	7	8	2	8	0	55
	5	8	6	0	0	3	35

B_6		COLUMN					
		1	2	3	4	5	RHS
	d_6	420	316	72	49	108	
	1	10	42	6	8	0	100
R	2	12	8	4	0	10	80
O	3	14	8	2	0	20	60
W	4	8	6	7	0	0	50
	5	10	6	1	0	3	35

B_7		COLUMN					
		1	2	3	4	5	RHS
	d_7	90	1811	430	3060	215	
	1	1	86	11	120	8	215
R	2	6	93	9	30	22	170
O	3	0	12	6	80	13	110
W	4	8	20	2	40	6	90
	5	4	30	12	16	18	80

B_8		COLUMN					
		1	2	3	4	5	RHS
	d_8	58	296	418	47	81	
	1	3	32	13	2	4	95
R	2	0	36	13	2	2	95
O	3	6	22	0	1	2	55
W	4	1	14	12	0	1	55
	5	3	16	30	4	0	60

Table A-III

Test Problem Number Three

c = (2250, 1725, 1025, 1945)

RHS = (-115, -116, 44, 83, 33)

ROW	COLUMN											
	A_1/A_2				B_1				B_2			
	1	2	3	4	1	2	3	4	1	2	3	4
OBJ(1)	64	36	18	34	44	-22	-62	- 6				
OBJ(2)	96	54	27	51					66	-33	-45	87
1	-42	-33	-45	-43	0	0	0	0	0	0	0	0
2	6	- 5	0	4	-46	-52	-64	-65	-46	-52	-51	-39
3	41	35	30	41	3	-14	-14	3	3	-14	-11	5
4	27	21	24	15	42	41	24	30	42	41	28	42
5	0	0	0	0	23	14	23	29	33	14	19	21

ROW	COLUMN											
	A_3/A_4				B_3				B_4			
	1	2	3	4	1	2	3	4	1	2	3	4
OBJ(3)	120	116	64	116	140	0	-32	52				
OBJ(4)	180	174	96	174					210	0	36	288
1	- 42	- 33	- 45	- 43	0	0	0	0	0	0	0	0
2	6	- 3	1	12	- 39	- 49	- 63	- 64	- 39	- 49	- 41	- 32
3	43	38	34	44	1	- 15	- 6	4	1	- 15	- 12	7
4	23	27	26	16	42	46	33	39	42	46	34	45
5	0	0	0	0	31	18	28	34	31	18	25	28

ROW	COLUMN											
	A_5/A_6				B_5				B_6			
	1	2	3	4	1	2	3	4	1	2	3	4
OBJ(5)	368	272	168	320	400	120	0	160				
OBJ(6)	552	408	252	480					600	180	180	672
1	- 42	- 33	- 45	- 43	0	0	0	0	0	0	0	0
2	10	4	5	11	- 38	- 44	- 59	- 57	- 38	- 44	- 42	- 32
3	48	38	34	50	7	- 10	- 6	7	7	- 10	- 7	13
4	30	25	27	22	49	48	33	38	49	48	37	47
5	0	0	0	0	32	21	32	32	32	21	27	28

ROW	COLUMN											
	A_7/A_8				B_7				B_8			
	1	2	3	4	1	2	3	4	1	2	3	4
OBJ(7)	220	168	80	180	260	148	48	148				
OBJ(8)	330	252	120	270					390	222	132	396
1	- 42	- 33	- 45	- 43	0	0	0	0	0	0	0	0
2	15	6	5	10	- 33	- 37	- 59	- 52	- 33	- 37	- 39	- 31
3	52	37	32	55	14	- 4	- 1	7	14	- 4	- 9	18
4	30	32	28	23	52	54	36	45	52	54	41	45
5	0	0	0	0	32	24	36	37	32	24	29	34

ROW	COLUMN											
	A_9/A_{10}				B_9				B_{10}			
	1	2	3	4	1	2	3	4	1	2	3	4
OBJ(9)	128	98	80	128	140	74	62	92				
OBJ(10)	192	147	120	192					210	111	108	258
1	- 42	- 33	- 45	- 43	0	0	0	0	0	0	0	0
2	16	7	10	19	- 29	- 40	- 50	- 50	- 29	- 40	- 33	- 29
3	51	45	39	57	11	- 6	- 1	13	11	- 6	- 4	22
4	39	30	36	31	52	53	42	43	52	53	41	56
5	0	0	0	0	36	30	40	40	36	30	32	37

Table A-IV

Test Problem Number Four

C = (2730, 1740, 1110, 1055)

RHS = (-121, -94, 68, 94, 35)

ROW	COLUMN											
	A_1/A_2				B_1				B_2			
	1	2	3	4	1	2	3	4	1	2	3	4
OBJ(1)	66	22	10	- 2	60	16	36	- 2				
OBJ(2)	99	33	15	- 3					90	24	- 60	21
1	-34	- 54	- 53	- 45	0	0	0	0	0	0	0	0
2	13	11	- 2	- 7	- 45	- 50	- 47	- 37	- 45	- 50	- 58	- 50
3	38	40	37	27	16	1	- 1	- 17	16	1	- 3	3
4	16	14	23	24	40	39	30	28	40	39	24	40
5	0	0	0	0	19	18	36	25	19	18	17	14

ROW	COLUMN											
	A_3/A_4				B_3				B_4			
	1	2	3	4	1	2	3	4	1	2	3	4
OBJ(3)	216	88	68	32	144	64	148	36				
OBJ(4)	324	132	102	48					216	96	- 42	156
1	- 34	- 54	- 53	- 45	0	0	0	0	0	0	0	0
2	21	16	4	- 6	- 45	- 44	- 46	- 30	- 45	- 44	- 56	- 47
3	43	45	37	34	16	- 3	8	- 12	16	- 3	3	8
4	24	15	29	25	44	41	40	30	44	41	29	45
5	0	0	0	0	21	22	35	21	21	22	17	20

ROW	COLUMN											
	A_5/A_6				B_5				B_6			
	1	2	3	4	1	2	3	4	1	2	3	4
OBJ(5)	440	296	168	176	416	216	360	144				
OBJ(6)	660	444	252	264					624	324	60	396
1	- 34	- 54	- 53	- 45	0	0	0	0	0	0	0	0
2	19	20	6	- 1	- 41	- 43	- 40	- 32	- 41	- 43	- 50	- 44
3	46	49	41	36	20	4	8	- 9	20	4	1	11
4	24	22	27	32	49	45	38	31	49	45	30	46
5	0	0	0	0	24	21	39	28	24	21	24	20

ROW	COLUMN											
	A_7/A_8				B_7				B_8			
	1	2	3	4	1	2	3	4	1	2	3	4
OBJ(7)	224	180	120	136	236	152	212	156				
OBJ(8)	336	270	180	204					354	228	66	240
1	- 34	- 54	- 53	- 45	0	0	0	0	0	0	0	0
2	21	22	9	- 2	- 35	- 38	- 41	- 29	- 35	- 38	- 51	- 42
3	46	53	42	43	25	5	10	- 4	25	5	- 1	15
4	23	24	32	38	47	49	38	37	47	49	32	47
5	0	0	0	0	22	22	46	35	22	22	31	20

ROW	COLUMN											
	A_9/A_{10}				B_9				B_{10}			
	1	2	3	4	1	2	3	4	1	2	3	4
OBJ(9)	146	110	78	80	144	104	134	76				
OBJ(10)	219	165	117	120					216	156	93	195
1	- 34	- 54	- 53	- 45	0	0	0	0	0	0	0	0
2	25	27	13	4	- 38	- 40	- 36	- 23	- 38	- 40	- 42	- 36
3	49	54	48	44	23	10	11	- 6	23	10	10	20
4	33	28	31	37	58	53	47	34	58	53	36	53
5	0	0	0	0	29	29	45	33	29	29	27	[illegible]

Table A-V

Test Problem Number Five

c = (1970, 1285, 1270, 2040, 1615)

RHS = (-196, -148, 105, 170, 60)

ROW	COLUMN														
	A_1/A_2					B_1					B_2				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
OBJ(1)	46	20	12	50	30	- 10	34	46	34	18					
OBJ(2)	69	30	18	75	45						-15	51	-15	9	-39
1	49	-57	-64	-37	-54	0	0	0	0	0	0	0	0	0	0
2	14	- 6	11	- 2	11	-62	-52	-36	-42	-52	-62	-52	-54	-49	-70
3	36	43	31	42	28	- 7	7	0	1	- 5	- 7	7	5	- 2	4
4	22	30	28	22	30	42	40	37	24	46	42	40	28	40	39
5	0	0	0	0	0	22	22	22	34	20	22	22	16	14	14

ROW	COLUMN														
	A_3/A_4					B_3					B_4				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
OBJ(3)	128	88	96	108	104	36	80	204	116	92					
OBJ(4)	192	132	144	162	156						54	120	102	162	-30
1	-49	-57	-64	-37	-54	0	0	0	0	0	0	0	0	0	0
2	13	- 3	19	0	19	-59	-50	-29	-40	-50	-59	-50	-44	-44	-71
3	42	46	37	43	27	- 2	5	6	5	2	- 2	5	13	4	3
4	26	36	32	21	34	49	42	47	32	47	49	42	27	44	46
5	0	0	0	0	0	21	23	27	32	24	21	23	21	23	17

ROW	COLUMN														
	A_5/A_6					B_5					B_6				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
OBJ(5)	336	208	184	336	240	136	360	400	336	256					
OBJ(6)	504	312	276	504	360						204	540	252	324	144
1	-49	-57	-64	-37	-54	0	0	0	0	0	0	0	0	0	0
2	18	3	19	6	17	-56	-43	-29	-39	-48	-56	-43	-46	-45	-65
3	44	46	35	48	32	- 2	10	5	10	3	- 2	10	11	7	10
4	29	34	33	25	35	49	49	46	33	49	49	49	33	43	46
5	0	0	0	0	0	26	29	28	38	28	26	29	23	22	21

ROW	COLUMN														
	A_7/A_8					B_7					B_8				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
OBJ(7)	156	116	132	208	168	76	212	240	172	180					
OBJ(8)	234	174	198	312	252						114	318	186	282	114
1	-49	-57	-64	-37	-54	0	0	0	0	0	0	0	0	0	0
2	17	2	24	12	18	-56	-42	-29	-40	-44	-56	-42	-40	-42	-59
3	44	51	33	48	37	3	17	8	11	7	3	17	10	14	10
4	27	33	40	29	41	48	49	47	31	52	48	49	36	49	49
5	0	0	0	0	0	24	29	34	41	30	24	29	25	26	19

ROW	COLUMN														
	A_9/A_{10}					B_9					B_{10}				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
OBJ(9)	122	82	84	114	104	82	138	132	132	122					
OBJ(10)	183	123	126	171	156						123	207	135	147	102
1	-49	-57	-64	-37	-54	0	0	0	0	0	0	0	0	0	0
2	27	12	27	10	26	-48	-36	-26	-30	-44	-48	-36	-39	-38	-61
3	50	49	41	53	40	5	16	11	17	11	5	16	17	11	13
4	33	37	38	31	40	53	53	50	37	58	53	53	37	48	55
5	0	0	0	0	0	31	36	31	42	36	31	36	30	28	27

Table A-VI

Test Problem Number Six

C = (1275, 2725, 2130, 300, 900)

RHS = (-163, -153, 97, 151, 61)

ROW	COLUMN														
	A_1/A_2					B_1					B_2				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
OBJ(1)	20	74	54	-38	-14	0	8	38	48	44					
OBJ(2)	30	111	81	-57	-21						0	12	12	-39	21
1	-42	-44	-34	-51	-46	0	0	0	0	0	0	0	0	0	0
2	6	13	11	- 3	-12	-52	-59	-49	-42	-46	-52	-59	-42	-52	-57
3	26	39	22	23	38	- 4	1	- 2	10	8	- 4	1	2	-10	12
4	20	29	28	12	13	42	34	43	42	33	42	34	28	31	30
5	0	0	0	0	0	14	28	27	14	27	14	28	16	18	22

ROW	COLUMN														
	A_3/A_4					B_3					B_4				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
OBJ(3)	68	164	120	8	56	64	48	112	120	124					
OBJ(4)	102	246	180	12	84						96	72	132	- 6	126
1	-42	-44	-34	-51	-46	0	0	0	0	0	0	0	0	0	0
2	14	16	11	3	-10	-49	-54	-51	-42	-49	-49	-54	-38	-56	-55
3	24	37	24	29	46	- 3	3	7	14	17	- 3	3	11	- 2	16
4	21	32	29	21	24	47	33	43	38	38	47	33	36	35	31
5	0	0	0	0	0	21	30	29	20	25	21	30	13	22	29

ROW	COLUMN														
	A_5/A_6					B_5					B_6				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
OBJ(5)	208	440	368	32	136	184	240	304	352	376					
OBJ(6)	312	660	552	48	204						276	360	348	36	384
1	-42	-44	-34	-51	-46	0	0	0	0	0	0	0	0	0	0
2	15	19	18	4	- 5	-45	-55	-45	-38	-42	-45	-55	-39	-49	-50
3	30	42	31	31	46	- 1	9	6	14	17	- 1	9	11	- 4	20
4	23	38	31	20	22	46	39	47	45	40	47	39	37	34	33
5	0	0	0	0	0	23	37	30	23	32	23	37	20	22	29

ROW	COLUMN														
	A_7/A_8					B_7					B_8				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
OBJ(7)	124	260	184	68	108	112	136	184	236	232					
OBJ(8)	186	390	276	102	162						168	204	234	90	300
1	-42	-44	-34	-51	-46	0	0	0	0	0	0	0	0	0	0
2	20	24	16	10	0	-46	-53	-40	-36	-44	-46	-53	-40	-43	-48
3	32	44	30	38	52	- 2	10	8	17	21	- 2	10	15	- 3	24
4	21	41	34	20	21	53	40	45	48	47	53	40	41	35	40
5	0	0	0	0	0	23	37	33	30	34	23	37	23	26	34

ROW	COLUMN														
	A_9/A_{10}					B_9					B_{10}				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
OBJ(9)	90	152	126	50	74	86	100	116	144	124					
OBJ(10)	135	228	189	75	11						129	150	135	75	156
1	-42	-44	-34	-51	-46	0	0	0	0	0	0	0	0	0	0
2	22	26	24	11	- 2	-42	-52	-38	-31	-39	-42	-52	-33	-45	-41
3	34	50	38	37	54	5	17	12	17	21	5	17	14	3	25
4	31	44	35	28	31	51	44	50	54	45	51	44	41	40	5[illegible]
5	0	0	0	0	0	29	41	34	32	35	29	41	23	27	[illegible]

REFERENCES

Armstrong, R.D. and P. Sinha, "Improved Penalty Calculations for a Mixed Integer Branch-and-Bound Algorithm," School of Engineering, Department of Industrial Engineering and Operations Research and School of Business Administration, Department of General Business and Finance, Technical Report No. 8, University of Massachussets, Amherst, June 1973.

Balas, E., "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," Operations Research, Vol. 13, No. 4 (July-August, 1965), pp. 517-546.

Balas, E., "Discrete Programming by the Filter Method," Operations Research, Vol. 15, No. 5 (September-October, 1967), pp. 915-957.

Baumol, W. J. and R.E. Quandt, "Investment and Discount Rates Under Capital Rationing -- A Programming Approach," Economic Journal, Vol. 75, No. 298 (June, 1965), pp. 317-329.

Benichou, M., J.M. Gauthier, P. Girodet, G. Hentgès, G. Ribière, and O. Vincent, "Experiments in Mixed Integer Linear Programming," Mathematical Programming, Vol. 1, No. 1 (October, 1971), pp. 76-94.

Bernhard, R.H., "Some Problems in the Use of a Discount Rate for Constrained Capital Budgeting," Journal of Finance and Quantitative Analysis, Vol. 2, No. 2 (June, 1969), pp. 111-158.

Byrne, R.F., A. Charnes, W.W. Cooper, and K.O. Kortanek, "A Chance-Constrained Approach to Capital Budgeting with Portfolio-Type Payback Constraints and Horizon Posture Controls," Chapter 3 in R.F. Byrne, et al., (ed.), Studies in Budgeting, North-Holland, Amsterdam, 1971a.

Byrne, R.F., A. Charnes, W.W. Cooper, and K.O. Kortanek, "C^2 and LPU^2 Combinations for Treating Different Risks and Uncertainties in Capital Budgets," Chapter 4 in R.F. Byrne, et al.,(ed.), Studies in Budgeting, North-Holland, Amsterdam, 1971b.

Carleton, W.J., "Linear Programming and Capital Budgeting Models: A New Interpretation," Journal of Finance, Vol. XXIV, No. 5 (December, 1969), pp. 825-833.

Charnes, A., W.W. Cooper, and M.H. Miller, "Application of Linear Programming to Financial Budgeting and the Costing of Funds," Journal of Business, Vol. XXXII, No. 1 (January, 1959), pp. 20-46.

Davis, R.E., D.A. Kendrick, and M. Weitzman, "A Branch-and-Bound Algorithm for Zero-One Mixed Integer Programming Problems," Development Economic Report No. 69, Center for International Affairs, Harvard University, 1967.

Dean, J., Capital Budgeting, Columbia University Press, New York, 1951.

Driebeek, N.J., "An Algorithm for the Solution of Mixed Integer Programming Problems," Management Science, Vol. 12, No. 7 (March, 1967), pp. 576-587.

Driebeek, N.J., Applied Linear Programming, Addison-Wesley, Reading, Mass., 1969.

Everett, H., III, "Generalized Lagrange Multiplier Method for Solving Problems of Optimal Allocation of Resources," Operations Research, Vol. 11, No. 3 (May-June, 1963), pp. 399-417.

Fisher, I., The Theory of Interest, Macmillan, New York, 1930.

Garfinkel, R.S. and G.L. Nemhauser, Integer Programming, Wiley, New York, 1972.

Geoffrion, A.M., "Integer Programming by Implicit Enumeration and Balas' Method," SIAM Review, Vol. 9, No. 2 (April, 1967), pp. 178-190.

Geoffrion, A.M., "An Improved Implicit Enumeration Approach for Integer Programming," Operations Research, Vol. 17, No. 3 (May-June, 1969), pp. 437-454.

Geoffrion, A.M., and R.E. Marsten, "Integer Programming Algorithms: A Framework and State-of-the-Art Survey," Management Science, Vol. 18, No. 9 (May, 1972), pp. 465-491.

Geoffrion, A.M., "Lagrangean Relaxation and It's Uses in Integer Programming," Working Paper No. 195, Western Management Science Institute, UCLA, December, 1972.

Glover, F., "A Multi-Phase Dual Algorithm for the Zero-One Integer Programming Problem," Operations Research, Vol. 13, No. 6 (November-December, 1965), pp. 879-919.

Haldi, J., "Twenty-Five Integer Programming Test Problems," Working Paper No. 43, Graduate School of Busines, Stanford University, December, 1964.

Hillier, F.S., "Derivation of Probabilistic Information for the Evaluation of Risky Investments," Management Science, Vol. 9, No. 3 (April, 1963), pp. 443-457.

Hillier, F.S., "Supplement to the Derivation of Probabilistic Information for the Evaluation of Risky Investments," Management Science, Vol. 11, No. 3 (January, 1965), pp. 485-487.

Hillier, F.S., "Chance-Constrained Programming with 0-1 or Bounded Continuous Decision Variables," Management Science, Vol. 14, No. 1 (September, 1967), pp. 34-51.

Hillier, F.S., and G.J. Lieberman, Introduction to Operations Research, Holden-Day, San Francisco, 1967.

Hillier, F.S., The Evaluation of Risky Interrelated Investments, North-Holland, Amsterdam, 1969.

Hirshleifer, J., "On the Theory of Optimal Investment Decision," Journal of Political Economy, Vol. LXVI, No. 4 (August, 1958), pp. 329-352.

Klevorick, A.K., "Capital Budgeting Under Risk: A Mathematical Programming Approach," Econometric Research Program, Research Memorandum No. 89, Princeton University, September, 1969.

Lorie, J., and L.J. Savage, "Three Problems in Rationing Capital," Journal of Business, Vol. XXVIII, No. 4 (October, 1955), pp. 229-239.

Lusztig, P., and B. Schwab, "A Note on the Application of Linear Programming to Capital Budgeting," Journal of Finance and Quantitative Analysis, Vol. 3, No. 4 (December, 1968), pp. 426-431.

Mao, J.C.T., Quantitative Analysis of Financial Decisions, Macmillan, New York, 1969.

Markowitz, H.M., Portfolio Selection, Wiley, New York, 1959.

Naslund, B., Decisions Under Risk, Economic Research Institute, Stockholm, 1967.

Naslund, B., "A Model of Capital Budgeting Under Risk," Chapter 2 in R.F. Byrne, et al.,(ed.), Studies in Budgeting, North-Holland, Amsterdam, 1971.

Petersen, C.C., "Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R&D Projects," Management Science, Vol. 13, No. 9 (May, 1967), pp. 736-750.

Roy, B., R. Benayoun, and J. Tergny, "From S.E.P. Procedure to the Mixed Ophelie Program," in J. Abadie (ed.), Integer and Nonlinear Programming, North-Holland, Amsterdam, 1970.

Schrage, L., "Using Decomposition in Integer Programming," Naval Research Logistics Quarterly, Vol. 20, No. 3 (September, 1973), pp. 469-476.

Seppälä, Y., "Constructing Sets of Uniformly Tighter Linear Approximations for a Chance Constraint," Management Science, Vol. 17, No. 11 (July, 1971), pp. 736-749.

Tomlin, J.A., "Branch-and-Bound Methods for Integer and Non-Convex Programming," in J. Abadie (ed.), Integer and Nonlinear Programming, North-Holland, Amsterdam, 1970.

Tomlin, J.A., "An Improved Branch-and-Bound Method for Integer Programming," Operations Research, Vol. 19, No. 4 (July-August, 1971), pp. 1070-1075.

Tuan, N. Ph., "A Flexible Tree-Search Method for Integer Programming Problems," Operations Research, Vol. 19, No. 1 (January-February, 1971), pp. 115-119.

Weingartner, H.M., "Capital Budgeting of Interrelated Projects: Survey and Synthesis," Management Science, Vol. 12, No. 7 (March, 1966a), pp. 485-516. Also in Weingartner [1967].

Weingartner, H.M., "Criteria for Programming Investment Project Selection," Journal of Industrial Economics, Vol. 15, No. 1 (November, 1966b), pp. 65-76. Also in Weingartner [1967].

Weingartner, H.M., Mathematical Programming and the Analysis of Capital Budgeting Problems, Markham, Chicago, 1967.

Weingartner, H.M., "Some New Views on the Payback Period and Capital Budgeting Decisions," Chapter 5 in R.F. Byrne, et al., (ed.), Studies in Budgeting, North-Holland, Amsterdam, 1971.

10 SEP 80 682

Thesis 163641
R2545 Reardon
Decomposition of dual-angular integer programs, with applications to multi-stage capital budgeting.
10 SEP 80 DISPLAY 682

163641
Thesis
R2545 Reardon
Decomposition of dual-angular integer programs, with applications to multi-stage capital budgeting.

thesR2545
Decomposition of dual-angular integer pr
3 2768 001 01275 0
DUDLEY KNOX LIBRARY